PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ^{6} :
G06F 17/30

A1

(11) International Publication Number: WO 00/20993

(43) International Publication Date: 13 April 2000 (13.04.00)

(21) International Application Number: PCT/US99/20748

(22) International Filing Date: 8 September 1999 (08.09.99)

(30) Priority Data:
09/166,670 5 October 1998 (05.10.98) US

(71) Applicant: MOTOROLA INC. [US/US]; 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

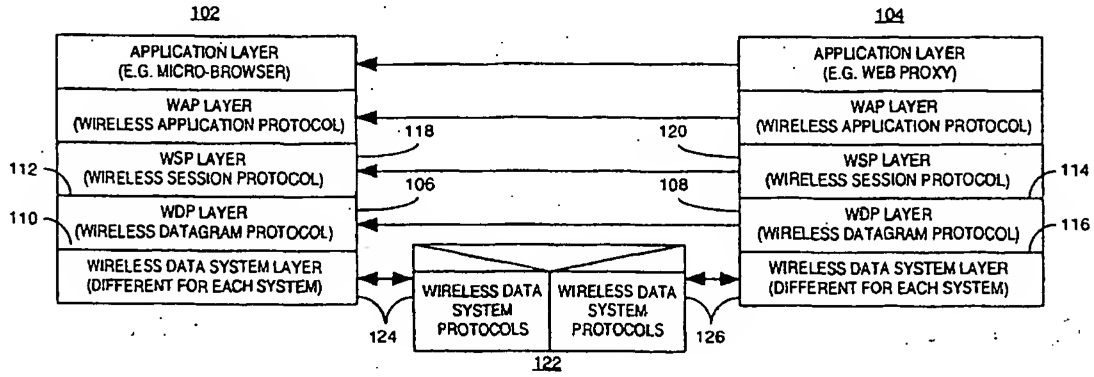
(72) Inventors: LOCKHART, Thomas, Wayne; 4580 Maple Crest, Vancouver, British Columbia V6J 4B4 (CA). WIEBE, Robert, John; 20 8451 Ryan Road, Richmond, British Columbia V7A 2E8 (CA). RICHARDSON, Christopher, John; 316 3480 Main Street, Vancouver, British Columbia V5V 2N2 (CA).

(74) Agents: NICHOLS, Daniel, K. et al.; Motorola Inc., Intellectual Property Dept., 1500 Gateway Boulevard, Boynton Beach, FL 33426-8292 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: METHOD AND APPARATUS FOR EFFICIENTLY ENCODING WIRELESS DATA PACKET HEADERS


102
104
APPLICATION LAYER
(E.G. MICRO-BROWSER)
APPLICATION LAYER
(E.G. WEB PROXY)
WAP LAYER
(WIRELESS APPLICATION PROTOCOL)
WAP LAYER
(WIRELESS APPLICATION PROTOCOL)
118
120
112
WSP LAYER
(WIRELESS SESSION PROTOCOL)
WSP LAYER
(WIRELESS SESSION PROTOCOL)
114
106
108
110
WDP LAYER
(WIRELESS DATAGRAM PROTOCOL)
WDP LAYER
(WIRELESS DATAGRAM PROTOCOL)
116
WIRELESS DATA SYSTEM LAYER
(DIFFERENT FOR EACH SYSTEM)
WIRELESS DATA SYSTEM PROTOCOLS
WIRELESS DATA SYSTEM PROTOCOLS
WIRELESS DATA SYSTEM LAYER
(DIFFERENT FOR EACH SYSTEM)
124
126
122

(57) Abstract

A method and apparatus for encoding wireless data packet headers for transmission of data between a client (102) and a server (104) operating in a data transmission system (100) includes providing a primary format code (202) to be placed within a data packet header field, the primary format code (202) determining when secondary format codes (206–212) are needed, providing conditionally one or more secondary format codes (206–212) to be placed within the data packet header field, the secondary format codes (206–212) determining the format for a set of data items (214) to be placed into a data packet data field, and providing conditionally each member of the set of data items (214) as indicated by the secondary format codes to be placed into the data packet data field, wherein a wireless data packet (200) is generated which can be transmitted between the client (102) and the server (104).

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

METHOD AND APPARATUS FOR EFFICIENTLY ENCODING WIRELESS DATA PACKET HEADERS

Field of the Invention

This invention relates in general to wireless communication systems and more specifically to a method for efficiently encoding wireless data packet headers.

Background of the Invention

One effort currently underway to make more efficient the use of radio communications systems for the transmission of data is the Wireless Application Protocol (WAP) Forum. The WAP Forum is currently in the process of defining a set of standards that allow efficient use of wireless systems to access the Internet's World Wide Web (the web) or other Internet applications. One part of the WAP suite of protocols is called Wireless Datagram Protocol or WDP. The purpose of WDP is to map from the application interfaces of various wireless data systems to one standard set of services that apply to any wireless data system. This permits the higher layers of WAP compatible communication systems to be independent of the particular type of wireless data system(s) being used to transport data between the remote clients and the data system's servers.

Certain information is required to be communicated between the wireless clients and the servers to provide the services defined by the WDP layer. A method of implementing WDP over various wireless data communication systems, is to add a WDP header containing the information required by the WDP layer to each packet. The prior art either includes all of the WDP information in each WDP packet, or devises and specifies variants of the WDP protocol for each of the particular wireless data systems that are carrying the WDP packets.

The problem addressed by this invention derives from the fact that some of the wireless communication systems already contain (or do not need) some of the information in the WDP header and this is wasteful of precious channel bandwidth. The invention disclosed herein describes a method for encoding the data to include only the required information which is not otherwise available. The invention also includes a method of efficiently encoding the information, even if it is needed and not currently available from the wireless data system.

Brief Description of the Drawings

FIG. 1 shows a simplified protocol stack showing how the Wireless Datagram Protocol (WDP) layer fits into a communication system in accordance with the present invention.

FIG. 2 shows a format for a WDP service data unit (WDP-SDU) in accordance with the present invention.

FIG. 3 shows a format for a WDP protocol data unit (WDP-PDU) in accordance with the present invention.

FIGs. 4-10 show flow charts for decoding WDP-PDUs in accordance with the preferred embodiment of the present invention.

FIGs. 11-14 show flow charts for encoding WDP PDUs in accordance with the preferred embodiment of the present invention.

FIG. 15 is an electrical block diagram representative of the client or server/proxy in accordance with the present invention.

FIG. 16 shows an example of the fully implicit form of the WDP-PDU format communicated in accordance with the present invention.

FIG. 17 shows an example of a WDP-PDU where the source address must be explicitly communicated in accordance with the present invention.

FIG. 18 shows an example of a WDP-PDU where the destination address must be explicitly communicated in accordance with the present invention.

FIG. 19 shows an example of a WDP-PDU where the source and destinations addresses as well as the segmentation and reassembly information must be explicitly communicated in accordance with the present invention.

FIG.20 shows a simplified protocol stack in accordance with an alternate embodiment of the present invention.

Description of the Preferred Embodiment

Referring now to the drawings and in particular to FIG. 1, there is shown a simplified protocol stack for Wireless Application Protocol (WAP) applications, which shows a client 102, typically a portable data communication device which accesses a wireless data system 100 using a wireless data network 122; and a server 104, typically a fixed communication device, which also accesses the wireless data system 100 using the wireless data network 122, and which also can access a wired data network, such as, but not limited to, the public switched telephone network (PSTN). The protocol stack for the client 102 includes an application layer, which provides user applications such as a micro-browser, a Wireless Application Protocol (WAP) layer which provides protocol conversion services, a Wireless Session Protocol (WSP) layer 118 which provides session layer services such as encryption and compression , a Wireless Datagram Protocol (WDP) layer 106 which will be described below, and a Wireless Data System layer 124 which provides the control of the physical hardware utilized for transmission and reception of data. The server 104 includes an application layer, which provides user applications such as a web proxy, a Wireless Application Protocol (WAP) layer, a Wireless Session Protocol (WSP) layer 120, a Wireless Datagram Protocol (WDP) layer 108, and a Wireless Data System layer 126 which provides the control of the physical hardware utilized for transmission and reception of data. The wireless data network 122 includes wireless data system protocols which control the distribution of data throughout the wireless data system 100.

The client 102 includes a WDP service access point 112 which provides a service that accepts WDP-SDU packets which are processed by the WDP layer 106 to generate WDP-PDU packets, also to be described below, which are passed through a wireless data system service access point 110 to the wireless data system layer 124. The WDP-PDU packets are passed across the wireless data network 122 to the peer entity, i.e. the server 104. The WDP layer 106 in the client 102, also processes the WDP-PDU packets passed through the wireless data system service access point 110 from the wireless data system layer 124 to the WDP layer 106 to generate WDP-SDU packets which are passed through the WDP service access point 112 to the next high layer.

The server 104 includes a WDP service access point 114 which provides a service that accepts WDP-SDU packets which are processed by the WDP layer 108 to generate WDP-PDU packets which are passed through a wireless data system service access point 116 to the wireless data system layer 126. The WDP-PDU packets are also passed across the wireless data network 122 to the peer entity, i.e. client 102. The WDP layer 108 in the server 104, also processes the WDP-PDU packets passed through the wireless data system service access point 116 from the wireless data system layer 126 to the WDP layer 108 to generate WDP-SDU packets which are passed through the WDP service access point 114 to the next high layer.

The WDP-SDU packet, to be described below, includes a source address type, a source address, a source port number, an optional destination address type, an optional destination address, an optional destination port number, and some user data. At the peer entity, the WDP-SDU packet must pass up either all of the information or at least the source address type, source address, source port number and user data. Further, the WDP layers 106 or 108 must be able to accept and pass large packets (e.g., up to 64 kilobytes) between the peers. To do this, the WDP layer 106 and WDP layer 108 must map the packets into the format required by the wireless data system 100, send them to the other side, and then map them back to the WDP-SDU format.

Examples of the client 102 include, but are not limited to, WAP portable handsets, GSM portable handsets, Pagewriter 2000™ data receivers, and DataTAC™ transceivers, such as manufactured by Motorola, Inc. Examples of the WAP server 104 include, but are not limited to, WAP proxy/servers, the DataTAC™ gateway, and the Wireless Messaging Gateway (WMG), such as manufactured by Motorola, Inc. Examples of the wireless data networks 122 include, but are not limited to, a GSM-SMS (Global System for Mobile Communications, Short Message Service), an Iridium®-SMS, an iDEN™-SMS, a DataTAC™ system, a Mobitex system, and a FLEX™ one-way or ReFLEX™ two-way data communication system.

Many of the wireless data systems do not provide services to communicate the source address, or either of the port numbers. Also, many wireless data systems do not support sufficiently large packets, so the WDP layer must segment the packets on one side and reassemble them on the other. To perform these tasks, the WDP layer must add data to each packet. The present invention describes an efficient method for communicating this extra information between the peer WDP layer 106 and WDP layer 108.

FIG. 2 shows a format for a WDP Service Data Unit, or WDP-SDU 200, in accordance with the present invention. A source address type field 202, is a single byte that identifies the type of source address present. This source address type implicitly identifies the length of the source address field. A source address field 204 identifies the computer system originating the WDP-SDU 200. A source port number field 206, is a two byte field that identifies the application program within the source system which originated the WDP-SDU 200. An optional destination address type field 208, is a single byte that identifies the type of destination address present. This destination address type implicitly identifies the length of the destination address field. An optional destination address field 210 identifies the computer system to which the WDP-SDU 200 is to be delivered. An optional destination port number field 212, is a two byte field that identifies the application program within the destination system to which the WDP-SDU 200 is to be delivered. A user data field 214, is from 1 to 65,536 bytes of user data, which is to be delivered to the destination. Note that all fields must be present when the WSP layer passes requests to the WDP layer, but the optional fields, 208, 210, and 212 may or may not be present when the WDP layer in the receiving entity passes the WDP-SDU indication to the WSP layer.

FIG. 3 shows the packet format for the WDP protocol data unit, or WDP-PDU 300 for short, in accordance with the present invention. Only the first byte of the WDP-PDU 300 is mandatory, all other bytes are optional, dependent upon the value of a format code 302. The first byte, (byte 0) of the WDP-PDU 300, always includes a WDP version code 304 and the format code 302. The low order nibble, 4 bits of byte 0, contain the version code 304 and indicate the version of the WDP protocol to which the WDP-PDU 300 conforms. By way of example, the initial version number is 0. The high order nibble, 4 remaining bits of byte 0, is the format code 302. The format code 302 has a number of different values and corresponding formats defined thusly:

Format 0x0 – reserved; This format code is not used.

Format 0x1 – debug; All remaining data is a debugging text message to be logged or otherwise handled by the receiving WDP layer.

Format 0x2 – implicit format; All WDP-PDU header bytes are NOT present. That is, the user data starts at byte 1, and the WDP layer is responsible for filling in the required WDP information implicitly. This can be done either by extracting the required information from the wireless data system protocol header, or by having (some of) the information configured into the receiving WDP entity, or by inferring the information from the recent history of the communication.

Format 0x3 – explicit source and destination codes; This format implies that both the "source address and port number code" and the "destination address and port number code" bytes are present, but not the "segment and re-assembly code" byte.

Format 0x4 – explicit source, destination and SAR codes; This format implies that all three of the "source address and port number code", "destination address and port number code", and "segment and re-assembly code" bytes are present.

Formats 0x5 through 0xE – reserved for future use.

Format 0xF – extended format; This format implies that the extended format code byte is present. No extended WDP-PDU formats are currently defined.

The particular values for the format codes described above are presented by way of example only. The only requirement is that the format codes are clearly specified and agreed to by the WDP layers in both the WDP client 102 and the WDP server 104.

The optional "extended format code" 306 is only present when the initial format code 302 is 0xF, as described above, and is 1 byte when present.

The optional "source address and port number code" byte, is only present in formats 0x3 and 0x4 and is divided into two nibbles. The high order nibble defines the optional source address type 308 as follows:

Address code = 0x0 – Indicates that an IPv4 address is present (4 bytes long).

Address code = 0x1 – Indicates that an IPv6 address is present (16 bytes long)

Address code = 0x2 – Indicates that a DataTAC LLI (Logical Link Identifier) address is present (4 bytes long)

Address codes 0x3 - 0x9 – Are reserved for other address types

Address codes 0xA - 0xD – Are reserved for other algorithmic compression mechanisms.

Address code = 0xE – Indicates that an extended address type code byte is present prior to the address.

Address code = 0xF – Indicates that the address is implicit, i.e. derived from the RF bearer protocol or pre-configured.

It should be noted that some address types have variable lengths, in which case the first byte of the address denotes the length of the address.

The low order nibble defines the optional source port number 310 as follows:

Port code = 0x0 – Indicates that a 1 byte extended port number is present.

Port code = 0x1 – Indicates that a 2 byte extended port number is present

Port codes 0x2 – 0x9 – Indicate that the port code is one of 8 (as determined by the exact code) pre-determined port numbers, which are port numbers defined for use with the WAP protocol suite.

Port codes 0xA-0xE – are reserved for future use.

Port code = 0xF – Indicates that the port number is implicit , i.e. derived from the RF bearer protocol or pre-configured.

The optional "destination address and port number code" byte is present in formats 0x3 and 0x4, and is defined the same as the " source address and port number code" byte, but refers to the optional destination address type 312 and optional destination port code 314.

The optional "segmentation and re-assembly code" byte is present in format 0x4 and is divided into three fields, described below:

The high order bit is a "more" flag 316. When set to 1 the bit indicates that this is not the last segment of the packet, i.e. there are more segments coming.

The next 3 most significant bits encode the segment number code 318 as follows:

Segment codes 0 - 4 are short segment numbers which are useful when a packet comprises 5 or fewer segments.

Segment code = 5 is reserved for future use.

Segment code = 6 indicates that a 1 byte extended segment number 334 is present.

Segment code = 7 indicates that a 2 byte extended segment number 334 is present.

The segment number code 318 possibly in combination with the optional extended segment number 334 indicates which segment (i.e. piece) of a multi-segment packet this is.

The low order nibble (4 bits) encode the packet number code 320, as follows:

Packet number codes 0x0-0xB are short packet numbers which are useful when 12 packet numbers are sufficient.

Packet number code = 0xC is reserved for future use.

Packet number code = 0xD indicates that a 1 byte extended packet number 336 is present.

Packet number code = 0xE indicates that a 2 byte extended packet number 336 is present.

Packet number code = 0xF indicates that a 4 byte extended packet number 336 is present.

The packet number code 320, possibly in combination with the optional extended packet number 336, indicates which packet this segment belongs to.

The optional "extended source address type" code 322 is present when so indicated by the "source address type" 308 described above and is 1 byte when present. The format of this byte will be defined in the future, as needed.

The optional "source address" 324 field is present when so indicated by the "source address type" 308. Its length is from 1 to 16 or more bytes, and is implied from the type of address, also specified in the "source address type" 308.

The optional "source port number" 326 is present when so indicated by the "source port code" 310. Its length is 1 or 2 bytes when present, is also indicated by the "source port code" 310.

The optional "extended destination address type" code 328 is present when so indicated by the "destination address type" 312 and is 1 byte when present. The format of this byte will be defined in the future, as needed.

The optional "destination address" field 330 is present when so indicated by the "destination address type" 312. Its length is from 1 to 16 or more bytes and is implied from the type of address, also specified in the "destination address type" 312.

The optional "destination port number" 332 field is present when so indicated by the "destination port code" 314. Its length is 1 or 2 bytes when present, and is also indicated by the "destination port code" 314.

The optional "extended segment number" 334 field is present when so indicated by the "segment number code" 318. Its length is 1 or 2 bytes when present, and is also indicated by the "segment number code" 318.

The optional "extended packet number" 336 field is present when so indicated by the "packet number code" 320. Its length is 1, 2, or 4 bytes when present, and is also indicated by the "packet number code" 320.

The "user data" field 338 is always present, and its length is from 1 to 65,536 bytes and is determined from information in the bearer protocol.

FIGs. 4-10 show flow charts for decoding WDP-PDUs in accordance with the preferred embodiment of the present invention. In the WDP-PDU decoding process, WDP-PDUs received from the RF network are translated to WDP-SDUs to be passed up to the higher layers of the protocol stack.

Beginning with FIG. 4, the WDP-SDU encoding process starts at step 402. At step 404 the WDP-PDU packet is received at either the WDP client 102 or the WDP server 104 depending upon which device originated the transmission and which device is receiving the transmission. The receiving device signal processor sets the "next data pointer" 1560 to the second byte of the WDP-PDU 300, at step 406. The first byte of the WDP-PDU 300 provides the version code. When the version code is not equal to zero (0), at step 408, indicating the version code is not a valid version code, the signal processor logs the version code as designating an unrecognized format and the WDP-PDU packet 300 is discarded at step 410, after which signal processing stops at step 422. When the version code is equal to zero (0), at step 408, the signal processor selects the high order nibble of the first byte-of the WDP-PDU 300 which provides the format code 302. When the signal processor determines that the format code 302 is equal to "debug", at step 412, the signal processor logs the message starting at the next data pointer 1560 and discards the WDP-PDU 300 packet, at step 414, after which signal processing stops at step 422. When the signal processor determines that the format code 302 is not equal to "debug", at step 412, the signal processor next checks whether the format code 302 is equal to "implicit", at step 416. When the signal processor determines that the format code 302 is equal to "implicit", at step 416, the signal processor begins to build a WDP-SDU 200 packet header using configured values for the source address type 202, source address 204, source port number 206, destination address type 208, destination address 210, and destination port numbers 212, at step 418. The signal processor also copies the remainder of the WDP-PDU packet 300 from the next data pointer 1560 (pointing at the WDP-PDU user data field 338) to the WDP-SDU user data field 214. The newly constructed WDP-SDU is then passed from the WDP layer to the next higher layer, at step 420, after which signal processing stops at step 422. When the signal processor determines the format code 302 is not equal to "implicit" at step 416, the signal processor next determines whether the format code 302 is equal to "explicit source and destination codes" code, at step 424. When the format code 302 is equal to "explicit source and destination codes" code, at step 424, the signal processor clears the SAR (segmentation and re-assembly) code present flag 1562 and increments the next data pointer 1560 by two (2), at step 426. The signal processor then jumps to the routine to decode the source address, at step 428, as will be described in further detail in FIG. 5. When the signal processor determines that the format code 302 does not equal to "explicit source and destination codes" code, at step 424, the signal processor determines whether the format code 302 is equal to the "explicit source destination and SAR codes" code, at step 430. When the signal processor determines the format code 302 is equal to the "explicit source destination and SAR codes" code, at step 430, the signal processor sets the SAR code present flag 1562, and increments the next data pointer 1560 by three, at step 432. The signal processor then jumps to the routine to decode the source address, at step 428, as will be described in further detail in FIG. 5. When the signal processor determines the format code 302 is not equal to the "explicit source destination and SAR codes" code, at step 430, the signal processor logs an undefined format code message and discards the WDP-PDU packet, at step 434, after which further processing by the signal processor is stopped at step 422.

Turning to FIG. 5, the decode source address routine 428 begins at step 504 when the signal processor initializes a buffer into which to construct the new WDP-SDU packet 1578, at step 504. The signal processor first determines whether the source address code 308 is equal to "implicit", at step 506. When the source address code is equal to "implicit", at step 506, the signal processor copies a pre-configured source address into the WDP-SDU packet under construction 1578, at step 508, after which the signal processor proceeds to the routine to decode source port number, at step 530, as will be described in FIG. 6 below. When the source address code 308 is not equal to "implicit", at step 506, the signal processor determines whether the source address code is equal to "extended", at step 510. When the source address code is equal to "extended", at step 510, the signal processor gets the extended source address type 322 from the next data pointer 1560 and increments the pointer by one (1), at step 512. The signal processor then determines whether the extended source address type 322 is within acceptable bounds, at step 516. When the extended source address type 322 is within acceptable bounds, at step 516, or the source address code is not equal to "extended", at step 510, the signal processor looks up the address length for the source address type (either normal or extended) in a pre-configured address length table. The signal processor next determines whether the source address length is negative indicating an undefined address type, at step 518. When the source address length is negative indicating an undefined address type, at step 518, the signal processor logs an indication that the address type is unrecognized, and discards the WDP-PDU and the WDP-SDU under construction, at step 520, after which further processing by the signal processor is stopped at step 522. When the source address length is not negative indicating the address type is defined, at step 518, the signal processor determines whether the address length is equal to zero (0), indicating that the length is variable, at step 524. When the address length is equal to zero (0), indicating that the length is variable, at step 524, the signal processor gets the extended source address length byte from the next data pointer 1560 and increments the pointer by one (1), at step 526, after which the signal processor proceeds to step 528. When the signal processor determines the address length is not equal to zero (0), indicating that the length is fixed, at step 524, the signal processor copies the number of bytes indicated by the source address length table from the next data pointer 1560 into the source address field 204 of the WDP-SDU under construction, and increments the pointer by the length, at step 528. The signal processor then jumps to the decode source port number routine, at step 530.

Turning to FIG. 6, the decode the source port number routine 530 begins at step 604 when the signal processor determines whether the source port code 310 indicates a one (1) byte extended port code 326, at step 604. When the source port code 310 indicates a one (1) byte extended port code, at step 604, the signal processor copies one (1) byte from the next data pointer 1560 into the source port number 206 field of the WDP-SDU packet under construction, and increments the next data pointer 1560 by one (1), at step 606. The signal processor then jumps to the decode destination address routine, at step 620. When the source port code 310 does not indicate a one (1) byte extended port code, at step 604, the signal processor determines whether the source port code 310 indicates a two (2) byte extended port code 326, at step 608. When the source port code 310 indicates a two (2) byte extended port code, at step 608, the signal processor copies two (2) bytes from the next data pointer 1560 into the source port number 206 of the WDP-SDU packet under construction, and increments the next data pointer 1560 by two (2), at step 610. The signal processor then jumps to the decode destination address routine, at step 620. When the signal processor determines the source port code 310 does not indicate a two (2) byte extended port code, at step 608, the signal processor determines whether the source port code 310 indicates an implied port code, at step 612. When the source port code 310 indicates an implied port code, at step 612, the signal processor copies a pre-configured source port number into the source port number 206 of the WDP-SDU packet under construction, at step 614. The signal processor then jumps to the decode destination address routine, at step 620. When the source port code 310 does not indicate an implied port code, at step 612, the signal processor maps the source port code 310 into a fixed source port number and copies the fixed source port number into the source port number 206 field of the WDP-SDU packet under construction, at step 616. The signal processor then determines whether the source port number 206 just looked up indicates an undefined source port code 310, at step 618. When the source port number 206 just looked up does not indicate an undefined source port code 310, at step 618, the signal processor then jumps to the decode destination address routine, at step 620. When the source port number 206 just looked up indicates an undefined source port code 310, at step 618, the signal processor logs an indication that the source port code 310 is unrecognized, and the WDP-PDU and the WDP-SDU being constructed are discarded, at step 622, after which further processing by the signal processor is stopped at step 624.

Turning to FIG. 7, the decode destination address routine 620 begins at step 704 when the signal processor decodes the destination address using the same process as described above for the source address, using instead codes and fields defined for the destination addresses. The processor then jumps to the decode destination port number routine, at step 706. The decode destination port number routine starts at step 708, where the signal processor decodes the destination port number using the same process as for decoding the source port number, using instead codes and fields defined for destination port numbers. The signal processor then determines whether the SAR code present flag 1562 is set, at step 710. When the SAR code present flag is set, at step 710, the signal processor jumps to the decode SAR information routine, at step 712. When the SAR code present flag is not set, at step 710, the signal processor copies the user data 338 from the next data pointer 1560 into the user data field 214 of the WDP-SDU packet under construction, getting the data length from the bearer protocol, at step 714. The signal processor then passes the completed WDP-SDU packet 200 to the next higher level, that is the signal processor sends to the next higher level an indication that the WDP-SDU packet is available, at step 716, after which further processing by the signal processor is stopped at step 718.

Turning to FIG. 8, the decode SAR information routine 712 begins at step 804 when the signal processor copies the more flag 316 from the WDP-PDU into a temporary location associated with the WDP-SDU packet under construction, at step 804. The signal processor then jumps to the decode segment number routine, at step 806. The signal processor immediately proceeds to determines whether the segment number code 318 indicates a one (1) byte extended segment number 334, at step 808. When the segment number code 318 indicates a one (1) byte extended segment number 334, at step 808, the signal processor copies one (1) byte from the next data pointer 1560 into a temporary segment number field associated with the WDP-SDU packet under construction, and increments the next data pointer 1560 by one (1), at step 810, after which the signal processor then jumps to the decode packet number routine, at step 820. When the segment number code 318 does not indicate a one (1) byte extended segment number 334, at step 808, the signal processor determines whether the segment number code 318 indicates a two (2) byte extended segment number 334, at step 812. When the segment number code 318 indicates a two (2) byte extended segment number 334, at step 812, the signal processor copies two (2) bytes from the next data pointer 1560 into a temporary segment number field associated with the WDP-SDU packet under construction and increments the next data pointer 1560 by two (2), at step 814, after which the signal processor then jumps to the decode packet number routine, at step 820. When the segment number code 318 does not indicate a two (2) byte extended segment number 334, at step 812, the signal processor then determines whether the segment number code 318 is in the short segment number range, at step 816. When the segment number code 318 is in the short segment number range, at step 816, the signal processor copies the short segment number from 318 into a temporary segment number field associated with the WDP-SDU packet under construction, at step 818, after which the signal processor then jumps to the decode packet number routine, at step 820. When the segment number code 318 is not in the short segment number range, at step 816, the signal processor logs an indication that the segment number code 318 is unrecognizable, and discards the WDP-PDU and the WDP-SDU packet under construction, at step 822, after which further processing by the signal processor is stopped at step 824.

Turning to FIG. 9, the decode packet number routine 820 begins at step 904 when the signal processor determines whether the packet number code 320 indicates a one (1) byte extended packet number 336. When the packet number code 320 indicates a one (1) byte extended packet number 336, at step 904, the signal processor copies one (1) byte from the next data pointer 1560 into a temporary packet number field associated with the WDP-SDU packet under construction, and increments the next data pointer 1560 by one (1), at step 906. The signal processor then jumps to the process segment routine, at step 920. When the packet number code 320 does not indicate a one (1) byte extended packet number 336, at step 904, the signal processor determines whether the packet number code 320 indicates a two (2) byte extended packet number 336, at step 908. When the packet number code 320 indicates a two (2) byte extended packet number 336, at step 908, the signal processor copies two (2) bytes from the next data pointer 1560 into the temporary packet number field of the WDP-SDU packet under construction, and increments the next data pointer 1560 by two (2), at step 910. The signal processor then jumps to the process segment routine, at step 920. When the packet number code 320 does not indicate a two (2) byte extended packet number 336, at step 908, the signal processor determines whether the packet number code 320 indicates a four (4) byte extended packet number 336, at step 912. When the packet number code 320 indicates a four (4) byte extended packet number 336, at step 912, the signal processor copies four (4) bytes from the next data pointer 1560 into the temporary packet number field of the WDP-SDU packet under construction, and increments the next data pointer 1560 by four (4), at step 914. The signal processor then jumps to the process segment routine, at step 920. When the packet number code 320 does not indicate a four (4) byte extended packet number 336, at step 912, the signal processor determines whether the packet number code 320 is in the short packet number range, at step 916. When the packet number code 320 is in the short packet number range, at step 916, the signal processor copies the short packet number from 320 into the temporary packet number field of the WDP-SDU packet under construction, at step 918. The signal processor then jumps to the process segment routine, at step 920. When the signal processor determines the packet number code 320 is not in the short packet number range, at step 916, the signal processor logs an indication that the packet number code 320 is unrecognized, and the WDP-PDU and the WDP-SDU packet under construction are discarded, at step 922, after which further processing by the signal processor is stopped at step 924.

Turning to FIG. 10, the process segment routine 920 begins at step 1004 when the signal processor adds the time the segment was received to the WDP-SDU buffer 1578, and saves the segment on a list of segments for the current packet number, source address type 202, source address 204, and source port number 206. The signal processor determines whether all segments for the current packet have been received, at step 1006. When all segments for the current packet have been received, at step 1006, the signal processor assembles all segments of the WDP-SDU into the WDP-SDU packet under construction, at step 1008. The signal processor then passes the completed WDP-SDU packet up to the next higher layer, i.e. sends an indication to the next higher layer that the WDP-SDU packet is available, at step 1010. When all segments for the current packet have not been received, at step 1006, and after the next higher layer has been informed that the WDP-SDU packet is available, at step 1010, the signal processor checks all saved segments to determine when any are too old, and if so, discards the old segments, at step 1012, after which further processing by the signal processor is stopped at step 1014.

FIGs. 11-14 show flow charts for encoding WDP PDUs in accordance with the preferred embodiment of the present invention. The encoding process translates WDP-SDUs received from the higher layers of the protocol stack (requests) into WDP-PDUs to be sent across the wireless data network to the peer entity.

Turning to FIG. 11, the WDP-PDU encoding process is described by way of example for a DataTAC system starts at step 1102. The signal processor sets the use implicit format flag 1566, at step 1104. The signal processor then determines the RF bearer (and receiver capability) to be used to deliver the WDP-SDU, at step 1106. This may be determined in one of at least three (3) methods, namely:

1) from the destination address type 208 passed with the WDP-SDU packet,

2) by mapping the destination address 210 (or sub-network implied by that address) to an RF bearer address, or

3) the RF bearer is implicit.

The latter of the methods is particularly relevant for clients operating on only one RF bearer. It is an implementation issue as to which of the three methods is ultimately used.

Based on the knowledge of the RF bearer and the receiver capability (such as by looking up a pre-configured table stored in the WDP client 102 or the WDP server 104) and the source address 204, the signal processor determines if the source address 324 can be implicit, at step 1108. By way of example, when the sender is the server/proxy side, the bearer is DataTAC, and the client supports multiple sources, the source address 324 cannot be implicit. When the receiver is the server/proxy side, and the bearer is DataTAC, then the source address 324 can be implicit, as the source address information is in the bearer's header. On the server/proxy side, the information about how to process the source address would either be implicit (if the server only interacts with clients with very specific capabilities) or built into a table indexed by the destination address information 208, 210, 212. When the encoding is not implicit, the signal processor clears the use implicit format flag 1566, and copies the source address type 202 into the WDP-PDU buffer source address type 308, and the source address 204 into the WDP-PDU buffer source address 324.

When the source port number 206 matches one of the constant WAP port numbers, the signal processor notes the associated port number code and saves it into the source port code 310 field of the WDP-PDU under construction, at step 1110, otherwise, based on knowledge of the RF bearer and the receiver capability (such as by looking up a pre-configured table stored in the WDP client 102 or the WDP server 104), and the source port, the signal processor determines when the source port can be implicit. If a constant port number cannot be used, a table lookup procedure can be used in a similar fashion to the source address 204, to determine if an implicit port number can be used. When the source port number 206 is less than 256, the signal processor notes in the source port code 310 of the WDP-PDU under construction that it will take one (1) byte to encode, otherwise the signal processor notes that the source port number 206 will require two (2) bytes to encode. When either the one (1) byte or two (2) byte extended source port code is required, the signal processor clears the use implicit format flag 1566 and copies the source port number 206 into the source port number 326 of the WDP-PDU under construction. The signal processor them jumps to the determine destination encoding routine, at step 1112.

Turning to FIG. 12, the determine destination encoding routine 1112 starts at step 1204 when the signal processor determines when the destination address 210 can be implicit using a similar lookup table technique as used for the source address 206. By way of example, when the sending device is the server/proxy, and the bearer is a DataTAC system, then the destination address 330 is in the bearer header and can thus be implicit. When the encoding is not implicit, the signal processor clears the use implicit format flag 1566 and copies the destination address type 208 to the destination address type 312 and the destination address 210 to the destination address of the WDP-PDU under construction. The signal processor next determines the destination port number 332, at step 1206, in a similar fashion to that used to determine the source port number 326 described above. The signal processor then clears the use implicit format flag 1566 when an extended port number is needed, and copies the extended port number to the WDP-PDU under construction. The signal processor next determines whether segmentation is required, at step 1208, by comparing the size of the user data 214 in the WDP-SDU packet to be sent with the maximum WDP-PDU packet that can be sent over the bearer, minus the WDP-PDU header required. The size of the WDP-PDU header required can be determined by the signal processor from the use implicit format flag 1566, and the data that may have been previously copied to the WDP-PDU buffer. The signal processor next determines if segmentation is required, at step 1210. When segmentation is required, at step 1210, the signal processor jumps to the determine SAR encoding routine, at step 1212. When segmentation is not required, at step 1210, the signal processor determines if the use implicit format flag 1566 is still set, at step 1214. When the use implicit format flag is still set, at step 1214, the signal processor builds a WDP-PDU packet with the implicit format, copies the user data 214 from the WDP-SDU packet to the user data field 338 of the WDP-PDU under construction, and passes the WDP-PDU packet to the bearer for transmission to the other side at step 1216, after which further processing by the signal processor is stopped at step 1218. When use implicit format flag is not set, at step 1214, the signal processor builds a WDP-PDU packet using the explicit source and destination codes format, and copies the source and destination codes, addresses, and port numbers into the WDP-PDU, as previously determined. The signal processor then copies the user data 214 from the WDP-SDU packet, and passes the WDP-PDU packet to the bearer for transmission to the other side at step 1216, after which further processing by the signal processor is stopped at step 1218.

Turning to FIG. 13, the determine SAR encoding routine 1212 starts at step 1304 when the signal processor determines the size of the packet number required which is largely an implementation issue based on the bearer and receiver type, and the method by which the sender generates packet numbers. Preferably, packet number sequences are distinct for each destination. The signal processor then selects the next available packet number, at step 1306, and adds the next available packet number to the packet number code 320 of the WDP-PDU under construction, and when necessary, copies the extended packet number 334 to the WDP-PDU packet under construction. The signal processor next determines the number of segments required by dividing the size of the WDP-SDU user data 214 by the WDP-PDU packet size for the selected bearer, minus the size of the WDP-PDU header constructed so far. The signal processor first assumes that short segment numbers will do to determine the header size. When more than the number of segments identifiable by short segment numbers (five) is needed, the signal processor increases the needed header size to account for one (1) byte segment numbers 332, and again determines the number of segments required. When the signal processor determines that more segments than can be identified with a one (1) byte segment number (256) are needed, the signal processor increases the header size to account for two (2) byte segment numbers 332, and recalculates the number of segments required. When the signal processor determines that more segments are required than can be identified with a two (2) byte segment number (65,536), then the signal processor determines something is wrong with the WDP-SDU since a WDP-SDU must be less than 65,536 bytes. The signal processor places a note about the problem in a log, aborts the WDP-SDU processing and stops. The signal processor builds a WDP-PDU header, at step 1310, using the format code 302 explicit source, destination, and SAR codes, copies the source address type 308, the source port code 310, the destination address type 312, the destination port code 314, the packet number code 320 and any addresses 324 and 330, and port numbers 326 and 332, and the expanded packet number 336, as required, and the signal processor sets the more flag

316. If the an extended segment number was determined previously to be required, then the segment number code 318 should also be copied to the WDP-SDU. The signal processor then jumps to the build multi-segments routine, at step 1312.

Turning to FIG. 14, the build multi-segments routine 1312 starts at step 1404 when the signal processor initializes a bytes left counter 1568 to the WDP-SDU user data size. The signal processor also initializes a user data count per PDU 1570 to the maximum WDP-PDU size minus the size of the WDP-PDU header just created above. The signal processor then initializes a next data pointer 1560 to the start of the user data 214 in the WDP-SDU, and finally initializes the segment number to zero (0). The signal processor next determines when the bytes left count 1568 is greater than the user data count per PDU 1570, at step 1406. When the bytes left count is not greater than the user data count per PDU, at step 1406, then the signal processor sets count 1572 to bytes left 1568 and clears the more flag 316 in the WDP-PDU header, at step 1408, and continues executing at step 1412. When the bytes left count 1568 is greater than the user data count per PDU 1570, at step 1406, the signal processor sets count 1572 to user data count per PDU 1572, at step 1410. The signal processor then builds a WDP-PDU packet by copying in the WDP-PDU header previously constructed, fills in the segment number (into either 318 if short segment numbers are being used or into 334 if extended segment numbers are being used, as determined previously in step 1308), and copies count 1572 bytes of data from the next data pointer 1560 into the user data 338 field of the WDP-PDU, at step 1412. The signal processor next passes the WDP-PDU to the bearer for transmission to the other side, at step 1414. The signal processor next determines when the bytes left count 1568 is equal to count 1572. When the bytes left count is not equal to count, the signal processor decrements the bytes left counter by count, increments the next data pointer 1560 by count, and increments the segment number by one (1), at step 1418, after which the process-flow returns to step 1406. When the bytes left count is equal to count, processing of the user data 214 by the signal processor is completed, and processing stops, at step 1420.

FIG. 15 is an electrical block diagram representative of the client or server/proxy in accordance with the present invention. When representative of the client 102, the client receiving device includes a transceiver 1502 which is used to transmit and receive WDP-PDU packets described above. Operation of the transceiver is well known to one of ordinary skill in the art. The input/output of the transceiver 1502 couples to a signal processor 1504 through an input/output (I/O) interface 1506. The signal processor is preferably a microcomputer, such as a member of the 68000 family manufactured by Motorola Inc.. The signal processor includes a central processing unit (CPU) 1508, a read only memory (ROM) 1510, a random access memory (RAM) 1512, and oscillator 1514, an annunciator driver 1516 and a display driver 1518. The I/O interface 1506 allows various peripheral devices to be connected to the signal processor including such devices as a code memory 1526, and user controls 1524. The code memory 1526 can be a programmable read only memory, or an electrically erasable programmable read only memory (EEPROM) which stores such information as the source and destination addresses, and the configuration tables needed by the processing routines. The user controls allow the user to control the operation of the client receiving device, providing such functions as means to reset the annunciators, means to input information to be transmitted, and means to recover information for display which has been received. It will be appreciated that many other functions can be provided by the user controls. The ROM 1510 stores the firmware programs which control the operation of the signal processor and the client receiving device as a whole. The firmware programs, include, but are not limited to: one or more application layer programs 1528, Wireless Application Protocol (WAP) routines 1530, Wireless Session Protocol (WSP) routines 1532, a source address decoder 1534, a source port number decoder 1536, a destination address decoder 1538, a destination port number decoder 1540, an SAR information decoder 1542, a segment number decoder 1544, a packet number decoder 1546, a segment processor 1548, a destination encoder 1550, an SAR information encoder 1552, a multi-segment processor 1554, additional Wireless Datagram Protocol (WDP) layer routines 1556, and a wireless system layer routine 1558. It will be appreciated that many other firmware programs can be stored as well. The function of many of the firmware programs listed were described above. The RAM 1512 stores variables utilized in the execution of the various firmware routines, including but not limited to: a next data pointer 1560, an SAR code present flag 1562, a more flag 1564, a use implicit format flag 1566, a bytes left counter value 1568, a user data count per PDU value 1570, a count value 1572, a segment number 1576, one or more WDP-SDU buffers 1578 and one or more WDP-PDU buffers 1580. The function of the various variables were described above. When a message is received, the user is alerted by way of an annunciator 1520 which is driven by the signal processor 1504 through an annunciator driver 1516. The annunciator can provide an audible alert, a tactile alert, or a visual alert, or any combination thereof. The user data 214 input by the client receiving device user, or received by the client receiving device can be displayed using a display 1522, such as an LCD display. The display is driven by the signal processor 1504 using a display driver 1518. The annunciator can be reset, user data 214 entered, and user data 214 recovered by way of user controls 1524 in a manner well known in the art.

FIG. 16 shows an example of the fully implicit form of the WDP-PDU format. The header consists of a single byte, containing two fields: the format code 302 field and the version code 304 field. The version code is equal to 0x0, indicating that this WDP-PDU conforms to first version of the WDP protocol. The format code is equal to 0x2, indicating that the source address type 202, source address 204, source port number 206, destination address type 208, destination address 210, and destination port number 212 are all implicit. The format code 0x2 also indicates that there is only one WDP-PDU segment for the WDP-SDU being communicated.

This variant of the WDP-PDU format could be used for example by a server that was sending a short WDP-SDU to a simple client over a DataTAC network, where both the server and the client were aware that the client only communicates with a single server, and its address type, address, and port number were pre-configured into the client. The server could know that its address was pre-configured into the client, because it only communicates with such clients. Alternately, the server could know its address is pre-configured into the client, because is has a table indexed by client address that has been pre-configured to indicate that the client is pre-configured only for this server. The destination address is carried in the DataTAC bearer service and is available to both the client and the server, so it can be implicit. The destination address type can be implicit, because both the server and the client are aware that a DataTAC network is being used, so the address type must be a DataTAC LLI. The destination port number can be implicit, because both the server and the client are aware that there is only one available port in the client.

This format could also be used by a simple DataTAC client when sending a small WDP-SDU to a server. In this case, the source address type must be a DataTAC LLI, which is known to both the client and the server.

The source address is carried by the DataTAC bearer and is available to both the client and the server. The source port number can be implicit because this is a simple client which only uses one port number and the server can know this (using the same mechanisms it would use to determine when sending WDP-SDUs to the client). The destination address type is the address of the type of the server, which is implicit for a DataTAC device and is optional when passing a WDP-SDU from the WDP layer to the WDP layer in the server. Similarly, the destination address and destination port number are also optional on WDP-SDUs sent from the WDP layer to the WSP layer in the server. Finally, the WDP-SDU is small enough to fit into a single DataTAC packet, so there is only one segment to this WDP-SDU and thus the segmentation and re-assembly information is not needed..

FIG. 17 shows an example of a WDP-PDU where the source address 204 must be explicitly communicated. This format could be used by a server that was sending a WDP-SDU to a DataTAC client where the server knew (from its configuration tables) that the client can communicate with many sources. Also, where the source port number 206 and destination port number 212 were both standard port numbers used for WAP applications (say the 2nd of the constant WAP port numbers) and the WDP-SDU user data 214 would have to be small enough to fit into a single DataTAC packet, i.e. less than 2000 bytes.

The format code 302, would be 0x3, indicating that the source address type 308, source port code 310, destination address type 312, and destination port code 314 were present. Assuming the server uses IPv4 addresses, then the source address type 308 would have a value of 0x0, indicating an IPv4 address 4 bytes long. This address type value would also indicate the presence of a 4 byte IPv4 address in the source address field 324. The source port code could be 0x3, indicating that the source port was the second of the constant WAP port numbers, and thus it was not necessary to include the source port number 326. Because the client is operating on the DataTAC system, the destination address is available in the bearer header and so the destination address type 312 can have a value of 0xF indicating that the destination address is implicit. The destination port number could be 0x3, indicating the second of the constant WAP port numbers, and thus it was not necessary to include the destination port number 332. Because the WDP-SDU is small enough to fit into a single DataTAC packet, the more flag 316, segment number code 318, and packet number code 320, as well as the associated extended segment number 334 and extended packet number 336 are also not needed. Also, because the format code 302 did not have the value 0xF, the extended format code 306 was not needed. Similarly, neither the extended source address type 322 nor the extended destination address type 328 were needed because they were not indicated by the source address type 308, or the destination address type 312.

This format would be useful, for example, in a system with a DataTAC client including an Internet world wide web micro-browser and a WAP proxy.

FIG. 18 shows an example of a WDP-PDU where the destination address 210 must be explicitly communicated. This format would be used by a DataTAC client that was sending a WDP-SDU to a WAP world wide web proxy. Also, where the source port number 206 and destination port number 212 were both standard port numbers used for WAP applications (say the 2nd of the constant WAP port numbers) and the WDP-SDU user data 214 would have to be small enough to fit into a single DataTAC packet, i.e. less than 2000 bytes

This is the reverse of the case shown in FIG. 17. That is this format might be used by a DataTAC world wide web client to send to a WAP proxy. In a similar fashion to the example shown in FIG. 17, the source address type 308, source port code 310, destination address type 312, and destination port code 314 are necessary due to the format selected, but the only other needed field is the destination address 330, as indicated by the destination address type 312.

Figure 19 shows an example of a WDP-PDU where most of the WDP-SDU information needs to be communicated explicitly and segmentation is required. This would be used when the server sends large WDP-SDUs to the client, but is unaware of the client's implicit capabilities, or where the client's capabilities do not include any implicit parameters. T he server using this technique, would set the format code 302 to 0x4, indicating explicit source, destination and SAR codes. The version code 304 would be set to 0x0, indicating that this WDP-PDU conforms to the first version of the WDP protocol specification. The source address type 308 would be set to 0x0, indicating that an IPv4 source address is present in the optional source address field 324. The source port code 310 would be set to 0x1, indicating that a two byte extended source port number 326 is present. The destination address type 312 would be set to 0x2, indicating that a 4 byte DataTAC LLI is present in the optional destination address field 330. The destination port code 314 would be set to 0x1, indicating that a two byte extended destination port number 332 is present. Assuming that five (5) or fewer segments would be required, the more flag 316 and segment number code 318 would be set appropriately for each segment. Finally, the packet number code would be set to 0xE, indicating that a two byte extended packet number 336 is present. Note that the optional extended format code 306, optional extended source address type 324, the optional extended destination address type 328, and the optional extended segment number 334 would not be present, because they were not needed nor indicated by the other format definitions.

FIG. 20 shows the protocol stack in accordance with an alternate embodiment of the present invention. In the alternate embodiment of the present invention, the server/proxy side WDP adaptation layer is placed in a logical component associated with the bearer. The advantage of this arrangement, is that WAP compliant server/proxies do not need to be modified to support bearers which include the WDP adaptation logical component. Figure 20 is very similar to figure 1, but differs from it in two ways. First there is a new component in the network, the logical component for WDP adaptation 2002. This component may be implemented as a distinct computer system, or it may be a software module integrated into the RF bearer 122, or the server proxy 104. This WDP adaptation component would implement the logic described above, except that it would receive and send WDP-SDUs across the interface point 2012. These WDP-SDUs would in turn be passed to or from another WDP adaptation layer 2006 in the WDP adaptation component 2002, which maps WDP-SDUs to the Internet standard User Datagram Protocol UDP. This is a straight forward mapping, as all of the necessary data for WDP-SDUs is contained in the UDP header and UDP supports large enough packets that segmentation and re-assembly mechanisms are not required. Also, this WDP-SDU to UDP mapping is defined in the WDP protocol specification. These UDP packets would be communicated to and from the server proxy 104 using the Internet Protocol (IP). UDP/IP is a standard mechanisms in common use on the Internet. The server/proxy 104 now needs to contain only a very simple WDP adaptation layer 2004 that maps WDP-SDUs to UDP packets.

While the preferred embodiments of the invention have been illustrated and described, it will be clear that changes, variations, substitutions and equivalents will occur to those skilled in the art without departing from the spirit and scope of the present invention as defined by the appended claims.

What is claimed is:

CLAIMS

1. A method for encoding wireless data packet headers for transmission of data between a client and a server operating in a data transmission system, said method comprising the steps of:

providing a primary format code to be placed within a data packet header field, the primary format code determining when secondary format codes are needed;

providing conditionally one or more secondary format codes to be placed within the data packet header field, the secondary format codes determining the format for a set of data items to be placed into a data packet data field; and

providing conditionally each member of the set of data items as indicated by the secondary format codes to be placed into the data packet data field,

wherein a wireless data packet is generated which can be transmitted between the client and the server.

2. The method for encoding wireless data packet headers of claim 1, wherein the secondary format codes distinguish between an implicit data item, one or more abbreviated data items, and a fully explicit data item.

3. A data communication device comprising:

a transceiver for transmitting and receiving wireless data packet which include a wireless data packet header and data;

a memory for storing a primary format code, one or more secondary format codes, and a set of data items; and

a signal processor, coupled to said transceiver, for decoding a wireless data packet which is received, and further for encoding a wireless data packet for transmission,

said signal processor encoding the wireless data packet for transmission by

recovering the primary format code to be placed within a data packet header field from said memory, the primary format code determining when secondary format codes are needed, recovering conditionally one or more secondary format codes to be placed within the data packet header field from said memory, the secondary format codes determining the format for the set of data items recovered from memory which are to be placed into a data packet data field; and

recovering conditionally each member of the set of data items as indicated by the secondary format codes to be placed into the data packet data field, thereby completing the encoding of a wireless data packet

said transceiver transmitting the encoded wireless data packet to another data communication device.

4. A data communication system comprising:

a first communication device comprising

a transmitter for transmitting a wireless data packet which includes a wireless data packet header and data,

a memory for storing a primary format code, one or more secondary format codes, and a set of data items, and

a signal processor, coupled to said transmitter, for encoding a wireless data packet for transmission to a second data communication device,

said signal processor encoding the wireless data packet for transmission by

recovering the primary format code to be placed within a data packet header field from said memory, the primary format code determining when secondary format codes are needed,

recovering conditionally one or more secondary format codes to be placed within the data packet header field from said memory, the secondary format codes determining the format for the set of data items recovered from memory which are to be placed into a data packet data field, and

recovering conditionally each member of the set of data items as indicated by the secondary format codes to be placed into the data packet data field, thereby completing the encoding of a wireless data packet; and

said second data communication device comprising a receiver for receiving the wireless data packet transmitted by said first data communication device

a memory for storing a set of data items, and

a signal processor, coupled to said receiver, for decoding the wireless data packet,

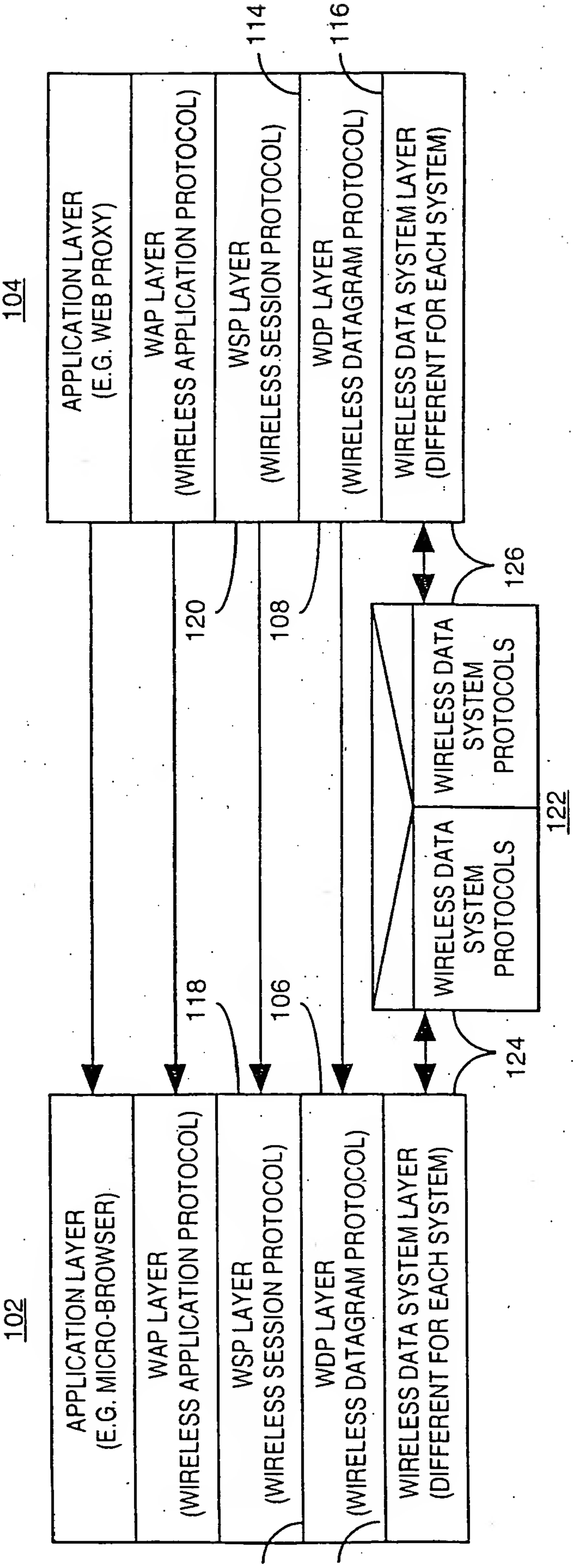
said signal processor decoding the wireless data packet by

recovering the primary format code from the data packet header field, the primary format code determining when secondary format codes are needed,

recovering the one or more secondary format codes from the data packet header field, the secondary format codes determining the format for the set of data items placed into the data packet data field, and

recovering each member of the set of data items as indicated by the secondary format codes from data packet data field, thereby completing the decoding of a wireless data packet, and

storing the set of data items recovered in the memory.


104
APPLICATION LAYER (E.G. WEB PROXY)
WAP LAYER (WIRELESS APPLICATION PROTOCOL)
WSP LAYER (WIRELESS SESSION PROTOCOL)
WDP LAYER (WIRELESS DATAGRAM PROTOCOL)
WIRELESS DATA SYSTEM LAYER (DIFFERENT FOR EACH SYSTEM)
114
116
120
108
126
WIRELESS DATA SYSTEM PROTOCOLS
WIRELESS DATA SYSTEM PROTOCOLS
122
118
106
124
102
APPLICATION LAYER (E.G. MICRO-BROWSER)
WAP LAYER (WIRELESS APPLICATION PROTOCOL)
WSP LAYER (WIRELESS SESSION PROTOCOL)
WDP LAYER (WIRELESS DATAGRAM PROTOCOL)
WIRELESS DATA SYSTEM LAYER (DIFFERENT FOR EACH SYSTEM)

100

FIG. 1

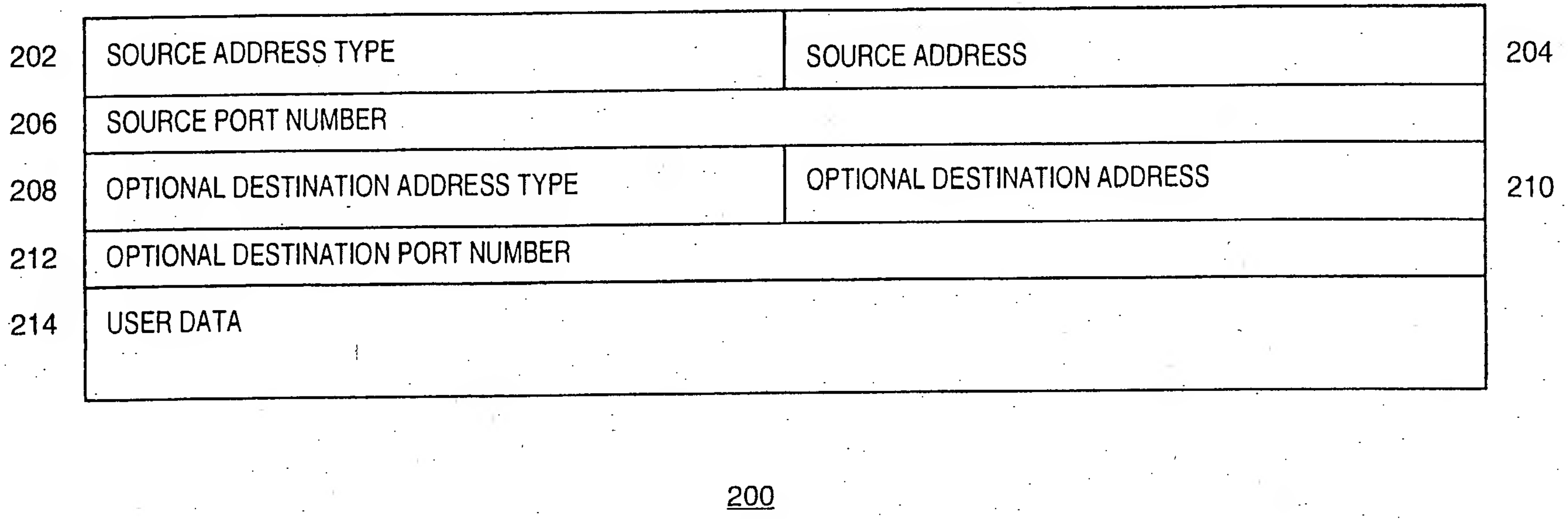
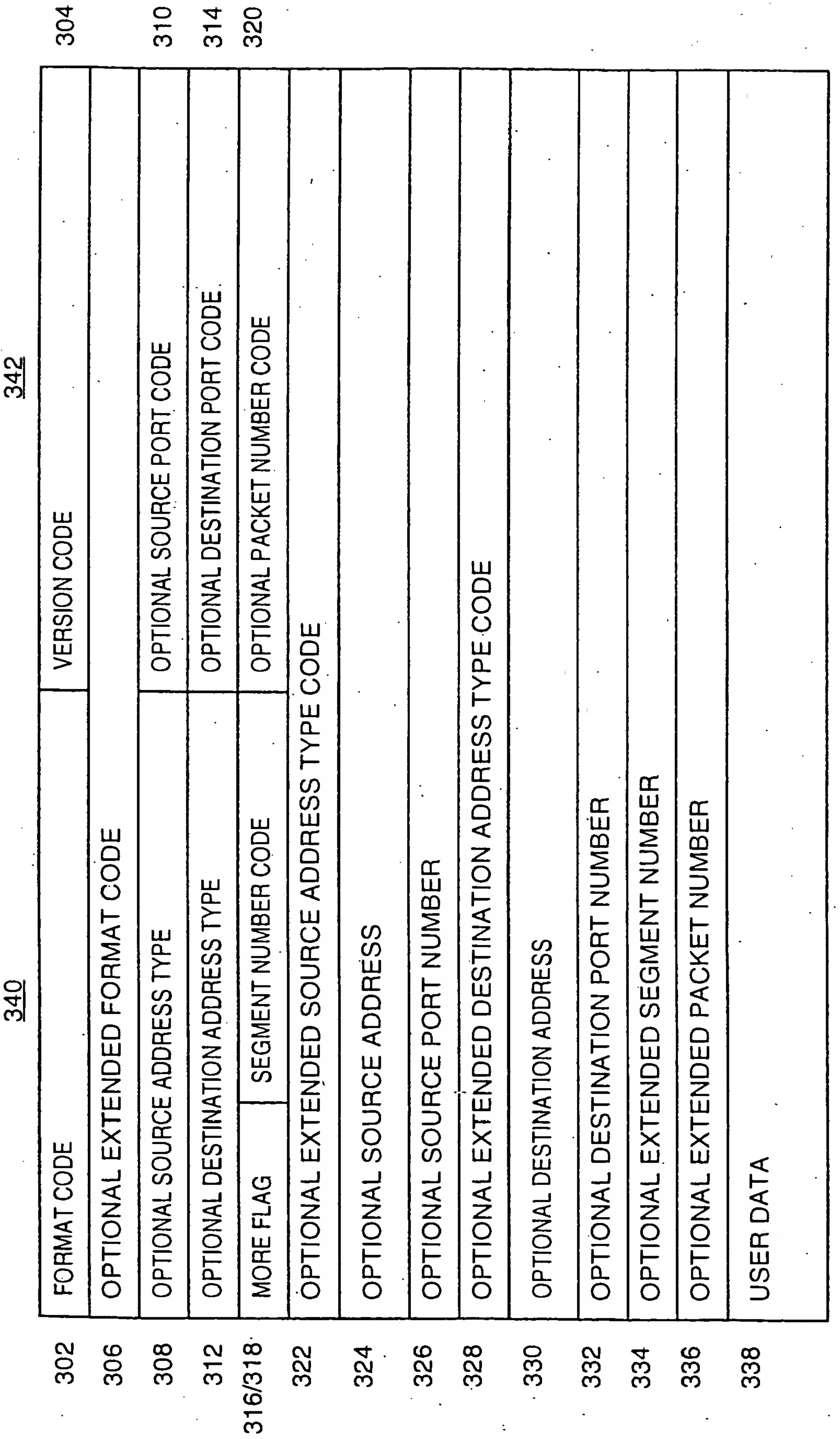
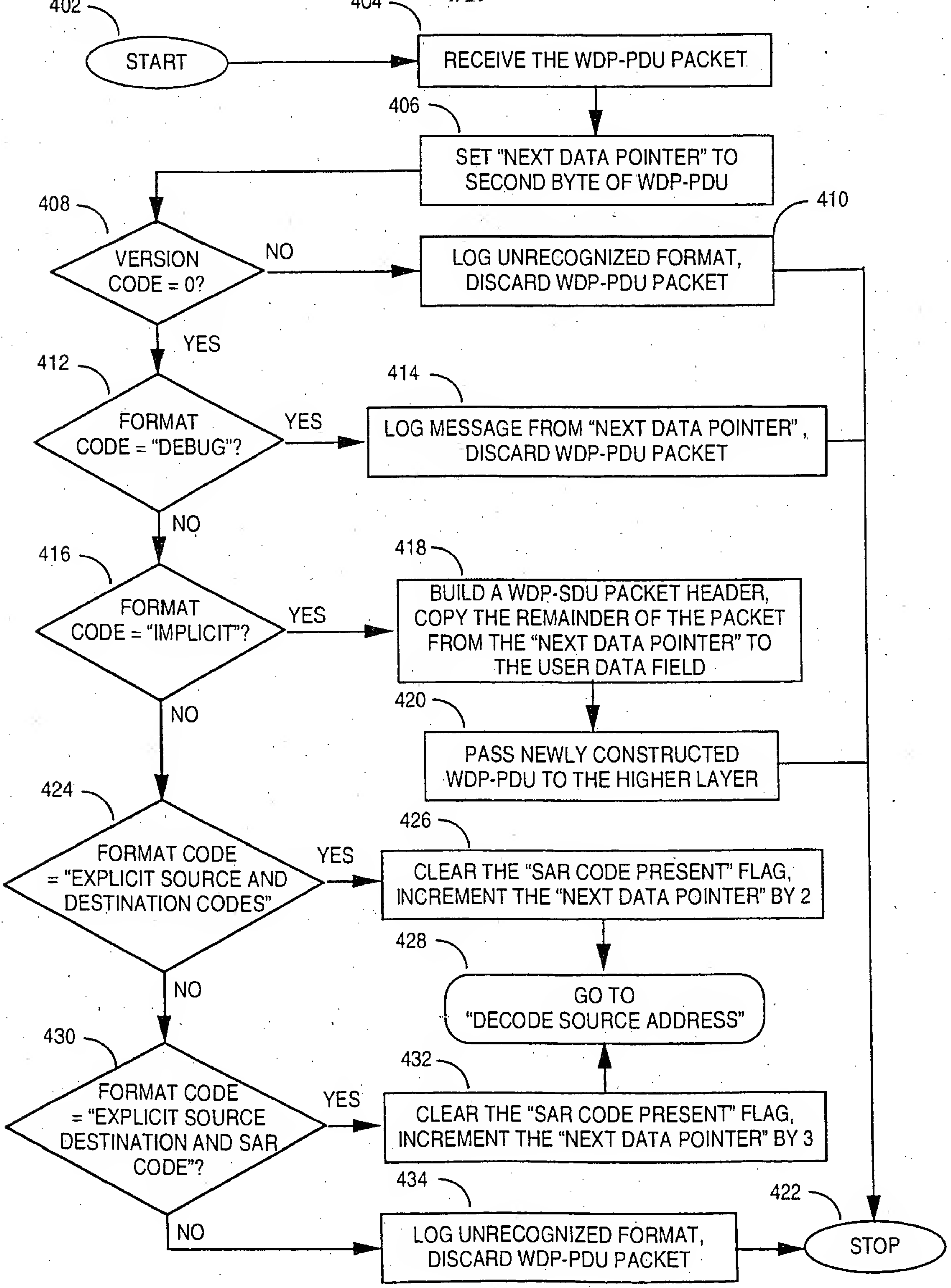

202
SOURCE ADDRESS TYPE
SOURCE ADDRESS
204
206
SOURCE PORT NUMBER
208
OPTIONAL DESTINATION ADDRESS TYPE
OPTIONAL DESTINATION ADDRESS
210
212
OPTIONAL DESTINATION PORT NUMBER
214
USER DATA
200

FIG. 2


340
342
302
FORMAT CODE
VERSION CODE
304
306
OPTIONAL EXTENDED FORMAT CODE
308
OPTIONAL SOURCE ADDRESS TYPE
OPTIONAL SOURCE PORT CODE
310
312
OPTIONAL DESTINATION ADDRESS TYPE
OPTIONAL DESTINATION PORT CODE
314
316/318
MORE FLAG
SEGMENT NUMBER CODE
OPTIONAL PACKET NUMBER CODE
320
322
OPTIONAL EXTENDED SOURCE ADDRESS TYPE CODE
324
OPTIONAL SOURCE ADDRESS
326
OPTIONAL SOURCE PORT NUMBER
328
OPTIONAL EXTENDED DESTINATION ADDRESS TYPE CODE
330
OPTIONAL DESTINATION ADDRESS
332
OPTIONAL DESTINATION PORT NUMBER
334
OPTIONAL EXTENDED SEGMENT NUMBER
336
OPTIONAL EXTENDED PACKET NUMBER
338
USER DATA

300

FIG. 3


402
START
404
RECEIVE THE WDP-PDU PACKET
406
SET "NEXT DATA POINTER" TO SECOND BYTE OF WDP-PDU
408
VERSION CODE = 0?
NO
410
LOG UNRECOGNIZED FORMAT, DISCARD WDP-PDU PACKET
YES
412
FORMAT CODE = "DEBUG"?
YES
414
LOG MESSAGE FROM "NEXT DATA POINTER", DISCARD WDP-PDU PACKET
NO
416
FORMAT CODE = "IMPLICIT"?
YES
418
BUILD A WDP-SDU PACKET HEADER, COPY THE REMAINDER OF THE PACKET FROM THE "NEXT DATA POINTER" TO THE USER DATA FIELD
420
PASS NEWLY CONSTRUCTED WDP-PDU TO THE HIGHER LAYER
NO
424
FORMAT CODE = "EXPLICIT SOURCE AND DESTINATION CODES"
YES
426
CLEAR THE "SAR CODE PRESENT" FLAG, INCREMENT THE "NEXT DATA POINTER" BY 2
428
GO TO "DECODE SOURCE ADDRESS"
NO
430
FORMAT CODE = "EXPLICIT SOURCE DESTINATION AND SAR CODE"?
YES
432
CLEAR THE "SAR CODE PRESENT" FLAG, INCREMENT THE "NEXT DATA POINTER" BY 3
NO
434
LOG UNRECOGNIZED FORMAT, DISCARD WDP-PDU PACKET
422
STOP

400

FIG. 4

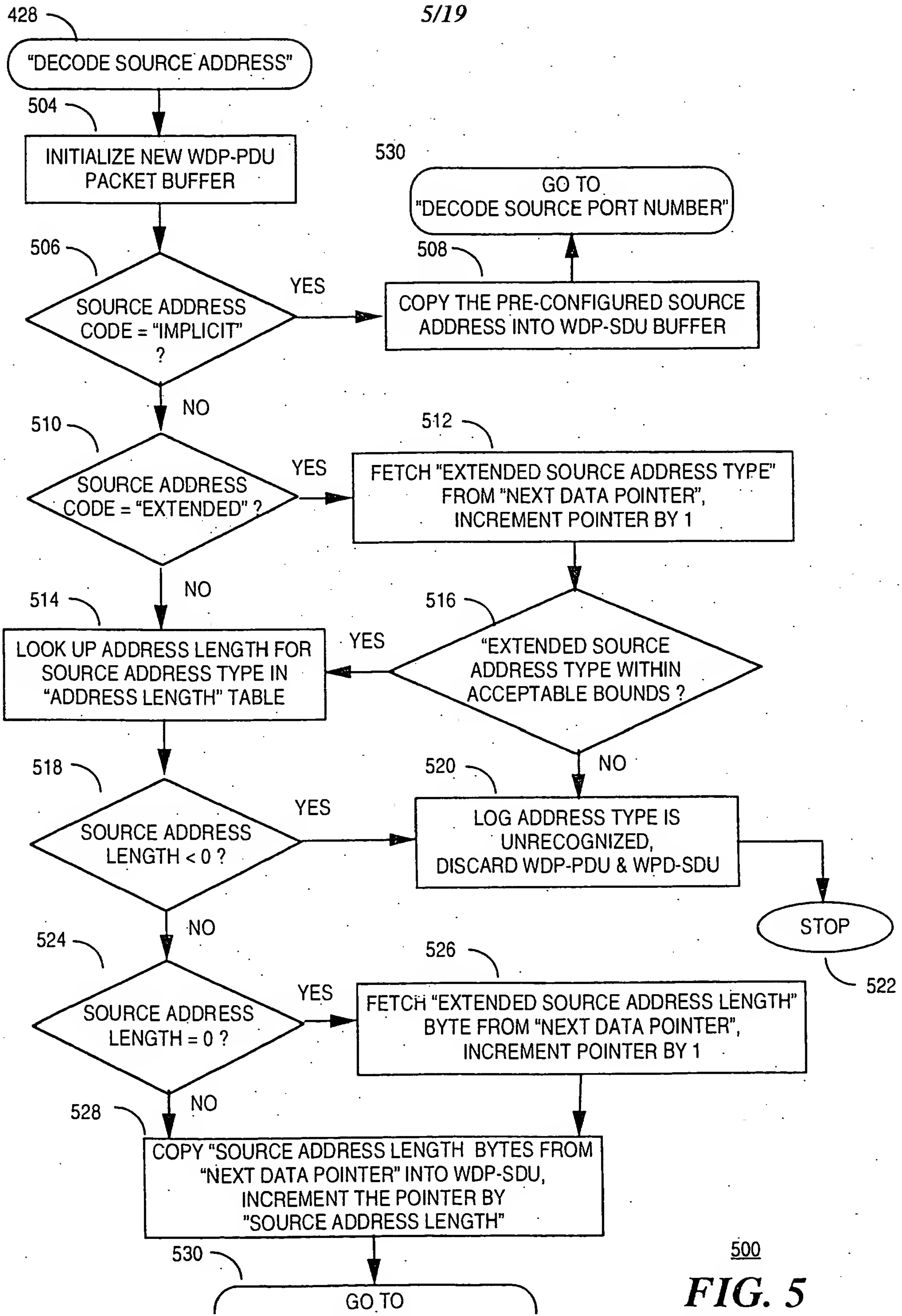
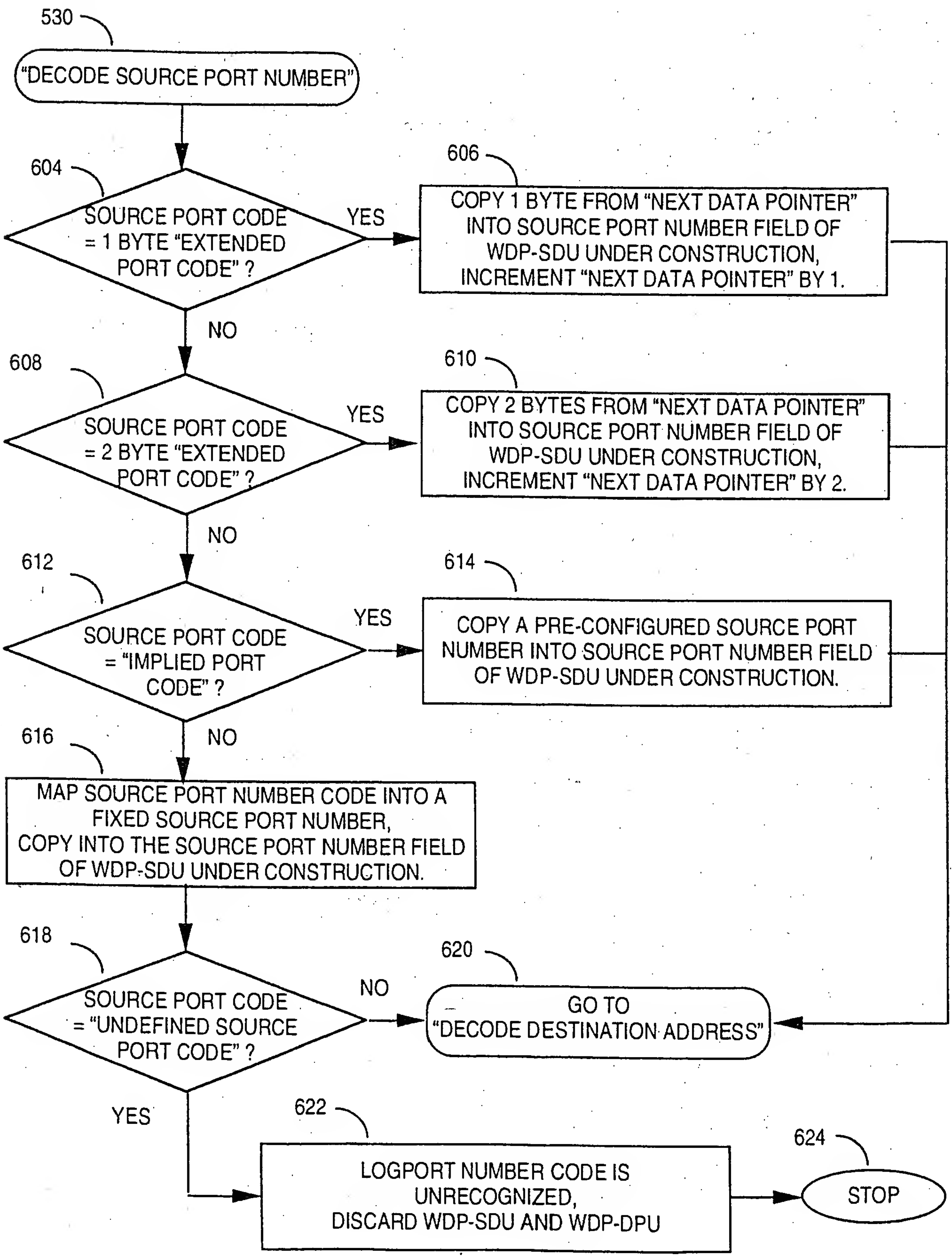
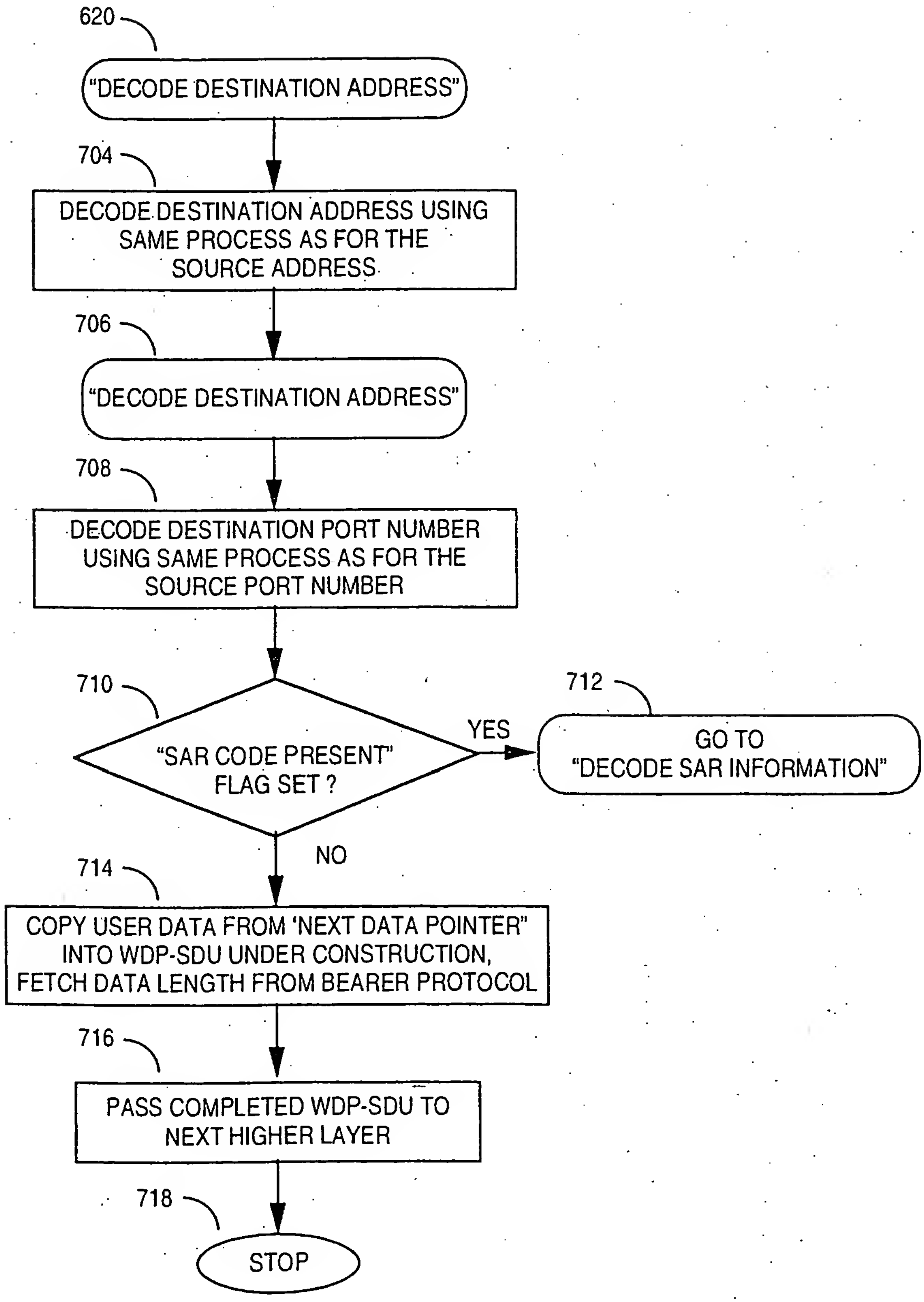

428
"DECODE SOURCE ADDRESS"
504
INITIALIZE NEW WDP-PDU PACKET BUFFER
530
GO TO "DECODE SOURCE PORT NUMBER"
506
508
YES
SOURCE ADDRESS CODE = "IMPLICIT" ?
COPY THE PRE-CONFIGURED SOURCE ADDRESS INTO WDP-SDU BUFFER
NO
510
512
YES
SOURCE ADDRESS CODE = "EXTENDED" ?
FETCH "EXTENDED SOURCE ADDRESS TYPE" FROM "NEXT DATA POINTER", INCREMENT POINTER BY 1
NO
514
516
YES
LOOK UP ADDRESS LENGTH FOR SOURCE ADDRESS TYPE IN "ADDRESS LENGTH" TABLE
"EXTENDED SOURCE ADDRESS TYPE WITHIN ACCEPTABLE BOUNDS ?
518
520
NO
YES
SOURCE ADDRESS LENGTH < 0 ?
LOG ADDRESS TYPE IS UNRECOGNIZED, DISCARD WDP-PDU & WPD-SDU
NO
STOP
524
526
522
YES
SOURCE ADDRESS LENGTH = 0 ?
FETCH "EXTENDED SOURCE ADDRESS LENGTH" BYTE FROM "NEXT DATA POINTER", INCREMENT POINTER BY 1
NO
528
COPY "SOURCE ADDRESS LENGTH BYTES FROM "NEXT DATA POINTER" INTO WDP-SDU, INCREMENT THE POINTER BY "SOURCE ADDRESS LENGTH"
530
GO TO
500

FIG. 5


530
"DECODE SOURCE PORT NUMBER"
604
SOURCE PORT CODE = 1 BYTE "EXTENDED PORT CODE" ?
YES
606
COPY 1 BYTE FROM "NEXT DATA POINTER" INTO SOURCE PORT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 1.
NO
608
SOURCE PORT CODE = 2 BYTE "EXTENDED PORT CODE" ?
YES
610
COPY 2 BYTES FROM "NEXT DATA POINTER" INTO SOURCE PORT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 2.
NO
612
SOURCE PORT CODE = "IMPLIED PORT CODE" ?
YES
614
COPY A PRE-CONFIGURED SOURCE PORT NUMBER INTO SOURCE PORT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION.
NO
616
MAP SOURCE PORT NUMBER CODE INTO A FIXED SOURCE PORT NUMBER, COPY INTO THE SOURCE PORT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION.
618
SOURCE PORT CODE = "UNDEFINED SOURCE PORT CODE" ?
NO
620
GO TO "DECODE DESTINATION ADDRESS"
YES
622
LOGPORT NUMBER CODE IS UNRECOGNIZED, DISCARD WDP-SDU AND WDP-DPU
624
STOP

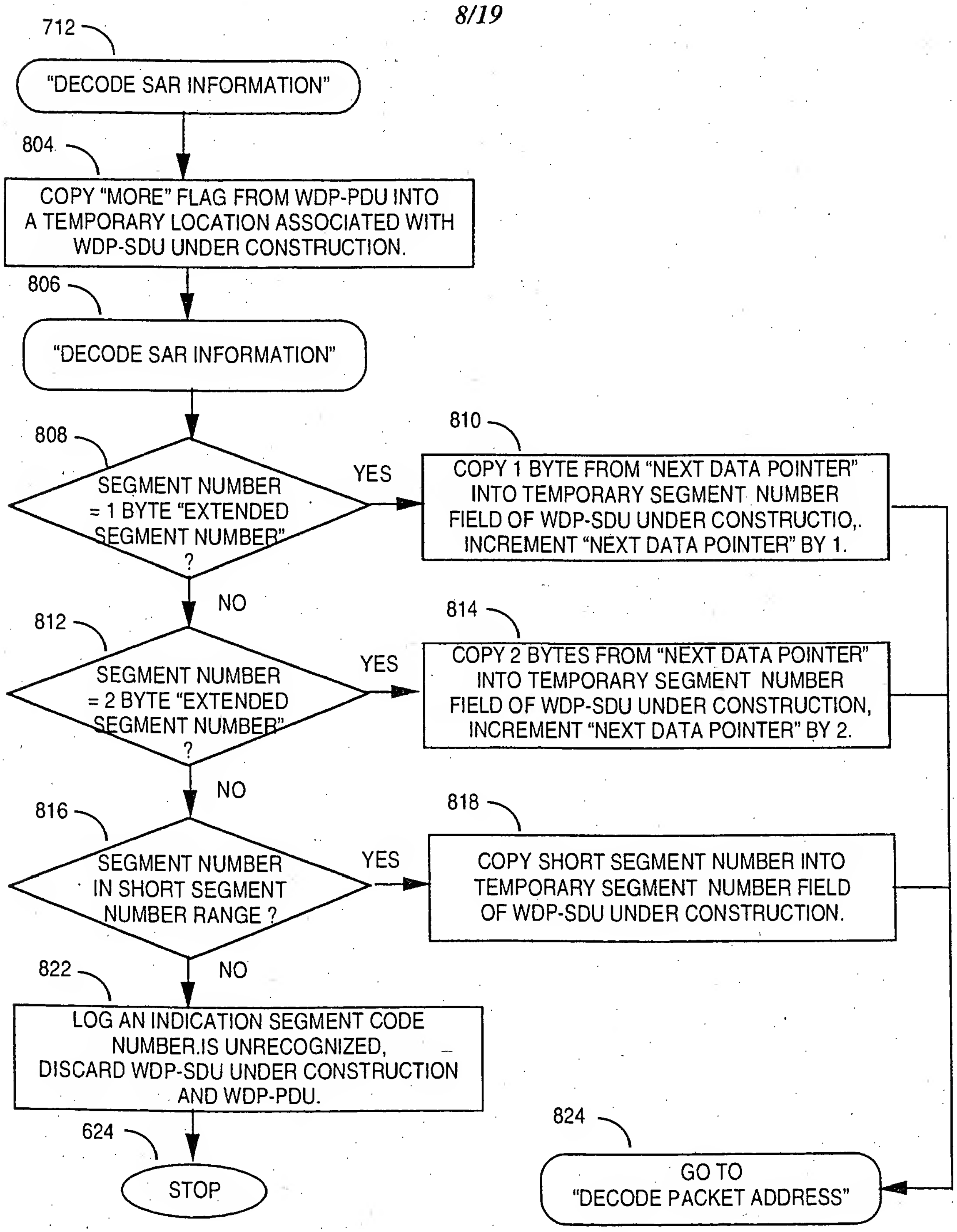
600

FIG. 6


620
"DECODE DESTINATION ADDRESS"
704
DECODE DESTINATION ADDRESS USING SAME PROCESS AS FOR THE SOURCE ADDRESS
706
"DECODE DESTINATION ADDRESS"
708
DECODE DESTINATION PORT NUMBER USING SAME PROCESS AS FOR THE SOURCE PORT NUMBER
710
"SAR CODE PRESENT" FLAG SET ?
YES
712
GO TO "DECODE SAR INFORMATION"
NO
714
COPY USER DATA FROM 'NEXT DATA POINTER" INTO WDP-SDU UNDER CONSTRUCTION, FETCH DATA LENGTH FROM BEARER PROTOCOL
716
PASS COMPLETED WDP-SDU TO NEXT HIGHER LAYER
718
STOP

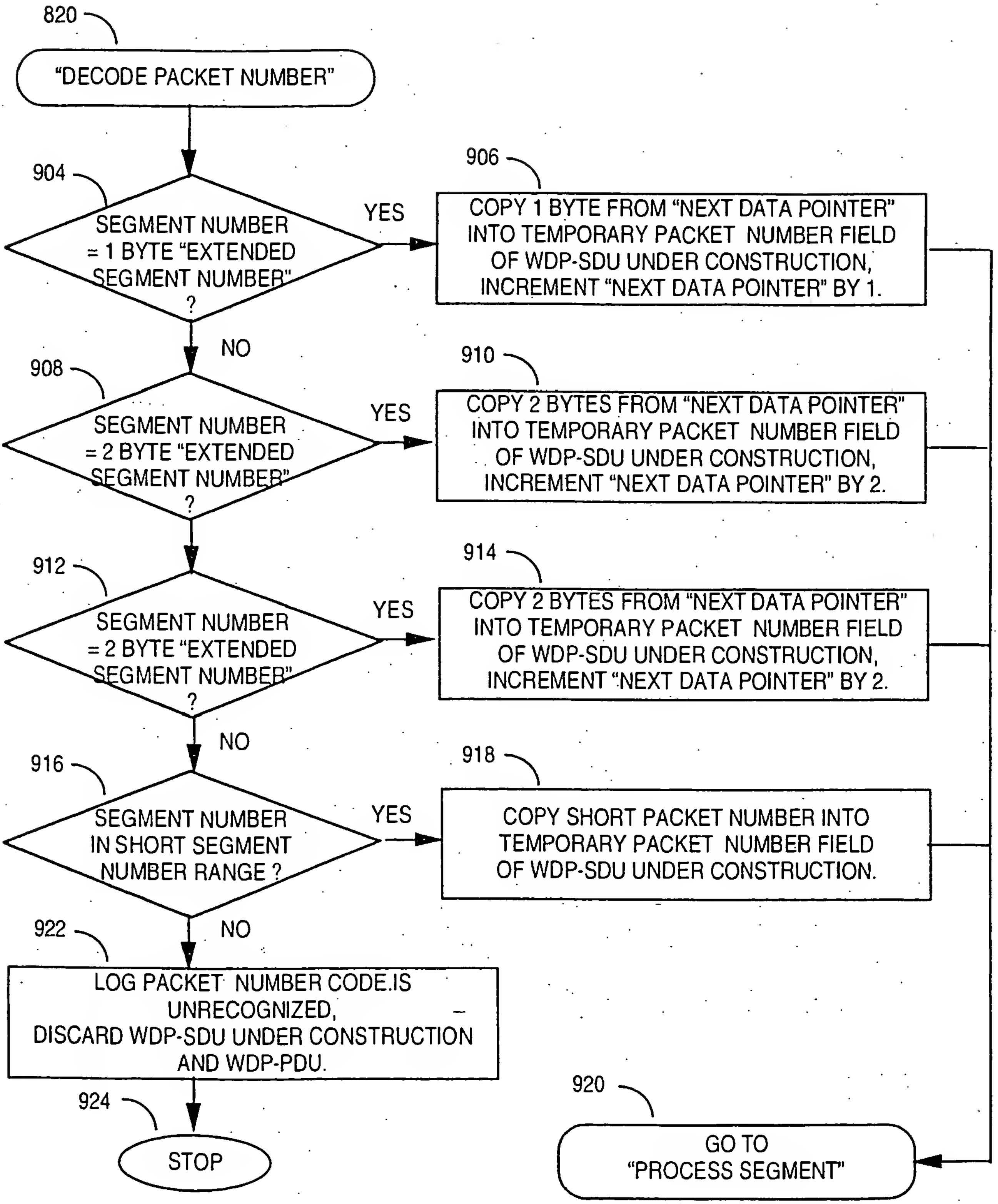
700

FIG. 7


712
"DECODE SAR INFORMATION"
804
COPY "MORE" FLAG FROM WDP-PDU INTO A TEMPORARY LOCATION ASSOCIATED WITH WDP-SDU UNDER CONSTRUCTION.
806
"DECODE SAR INFORMATION"
808
SEGMENT NUMBER = 1 BYTE "EXTENDED SEGMENT NUMBER" ?
YES
810
COPY 1 BYTE FROM "NEXT DATA POINTER" INTO TEMPORARY SEGMENT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTIO,. INCREMENT "NEXT DATA POINTER" BY 1.
NO
812
SEGMENT NUMBER = 2 BYTE "EXTENDED SEGMENT NUMBER" ?
YES
814
COPY 2 BYTES FROM "NEXT DATA POINTER" INTO TEMPORARY SEGMENT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 2.
NO
816
SEGMENT NUMBER IN SHORT SEGMENT NUMBER RANGE ?
YES
818
COPY SHORT SEGMENT NUMBER INTO TEMPORARY SEGMENT NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION.
NO
822
LOG AN INDICATION SEGMENT CODE NUMBER.IS UNRECOGNIZED, DISCARD WDP-SDU UNDER CONSTRUCTION AND WDP-PDU.
624
STOP
824
GO TO "DECODE PACKET ADDRESS"

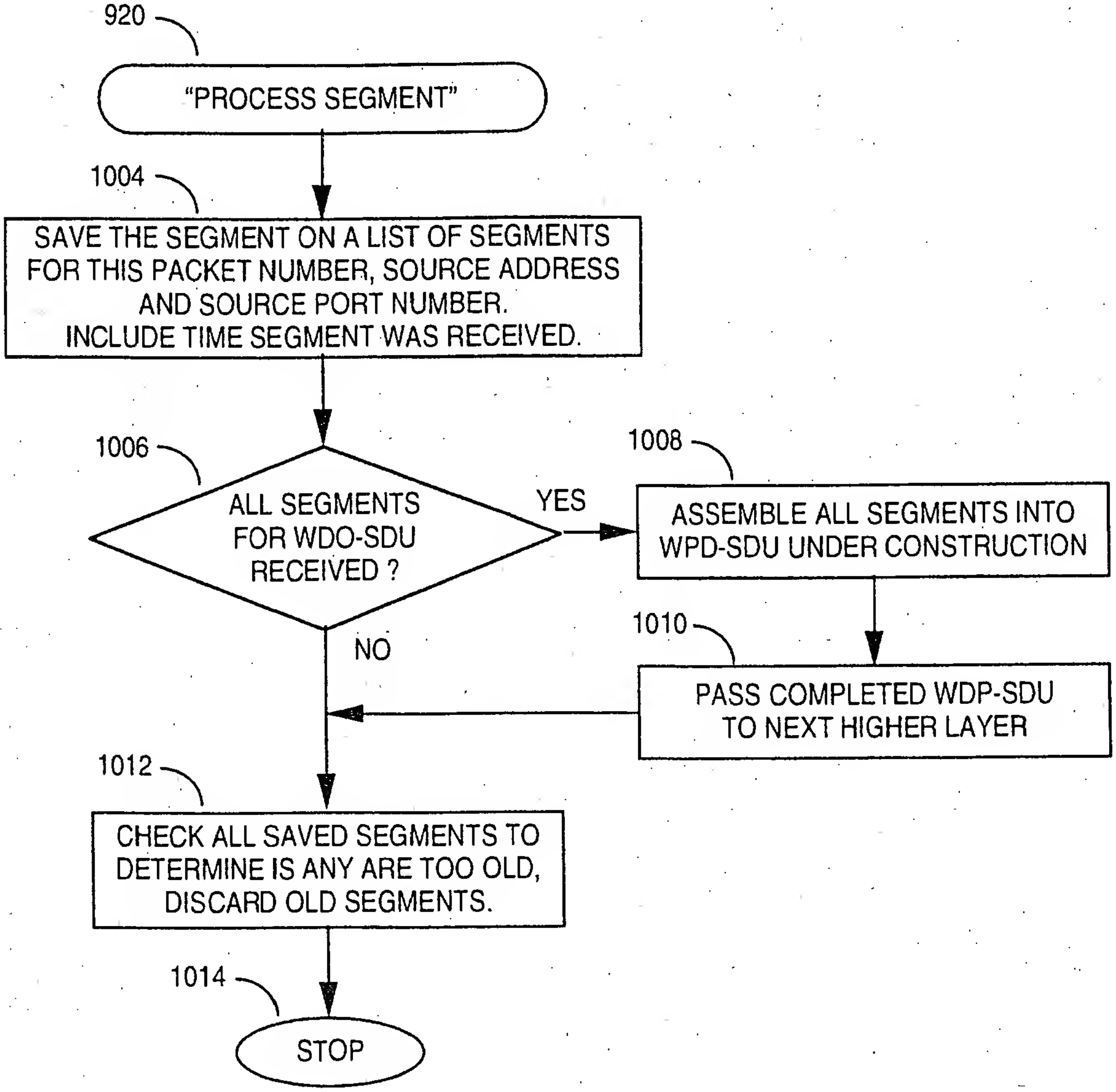
800

FIG. 8


820
"DECODE PACKET NUMBER"
904
SEGMENT NUMBER = 1 BYTE "EXTENDED SEGMENT NUMBER" ?
YES
906
COPY 1 BYTE FROM "NEXT DATA POINTER" INTO TEMPORARY PACKET NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 1.
NO
908
SEGMENT NUMBER = 2 BYTE "EXTENDED SEGMENT NUMBER" ?
YES
910
COPY 2 BYTES FROM "NEXT DATA POINTER" INTO TEMPORARY PACKET NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 2.
912
SEGMENT NUMBER = 2 BYTE "EXTENDED SEGMENT NUMBER" ?
YES
914
COPY 2 BYTES FROM "NEXT DATA POINTER" INTO TEMPORARY PACKET NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION, INCREMENT "NEXT DATA POINTER" BY 2.
NO
916
SEGMENT NUMBER IN SHORT SEGMENT NUMBER RANGE ?
YES
918
COPY SHORT PACKET NUMBER INTO TEMPORARY PACKET NUMBER FIELD OF WDP-SDU UNDER CONSTRUCTION.
NO
922
LOG PACKET NUMBER CODE.IS UNRECOGNIZED, DISCARD WDP-SDU UNDER CONSTRUCTION AND WDP-PDU.
924
STOP
920
GO TO "PROCESS SEGMENT"

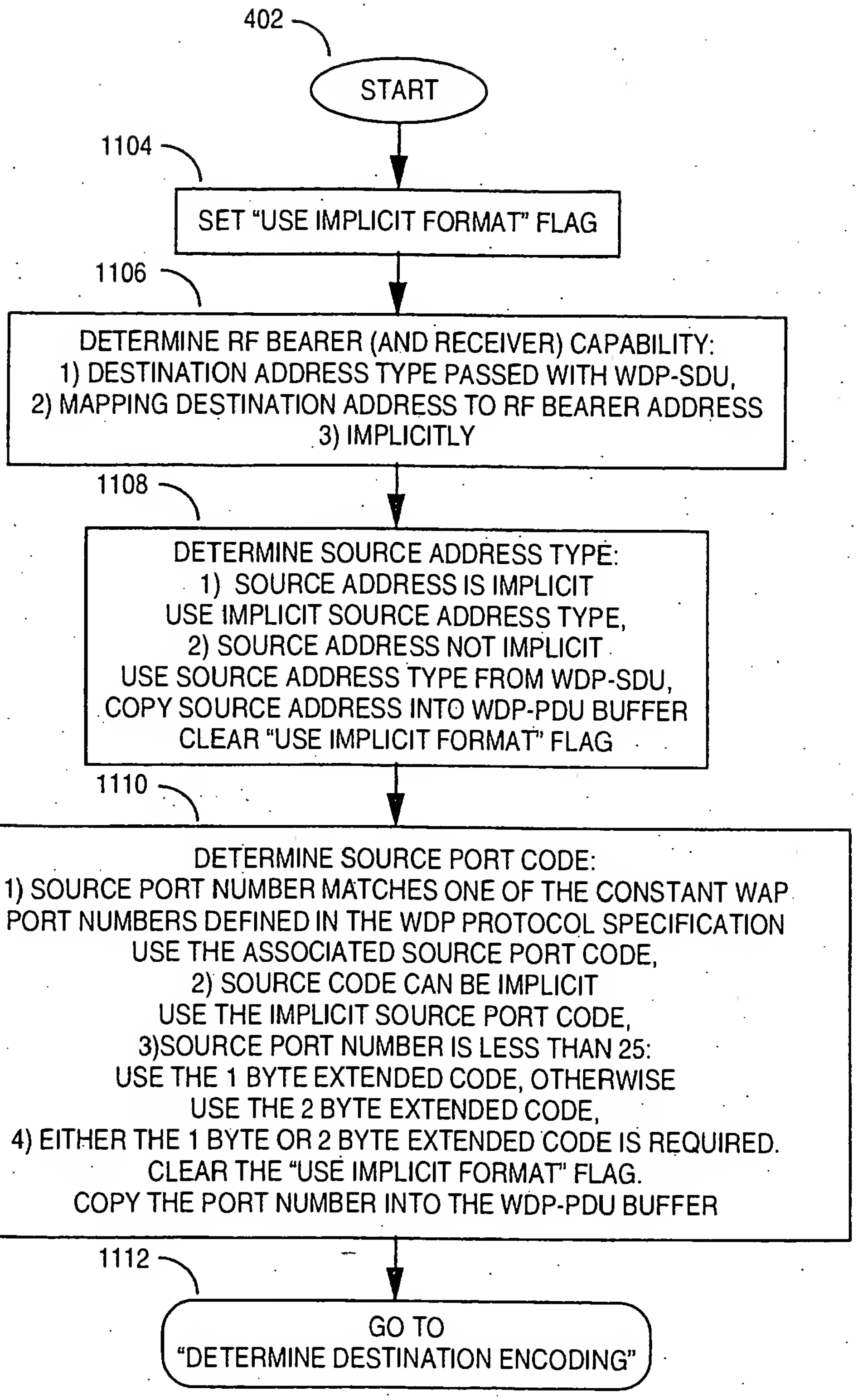
900

FIG. 9


920
"PROCESS SEGMENT"
1004
SAVE THE SEGMENT ON A LIST OF SEGMENTS FOR THIS PACKET NUMBER, SOURCE ADDRESS AND SOURCE PORT NUMBER. INCLUDE TIME SEGMENT WAS RECEIVED.
1006
ALL SEGMENTS FOR WDO-SDU RECEIVED ?
YES
NO
1008
ASSEMBLE ALL SEGMENTS INTO WPD-SDU UNDER CONSTRUCTION
1010
PASS COMPLETED WDP-SDU TO NEXT HIGHER LAYER
1012
CHECK ALL SAVED SEGMENTS TO DETERMINE IS ANY ARE TOO OLD, DISCARD OLD SEGMENTS.
1014
STOP

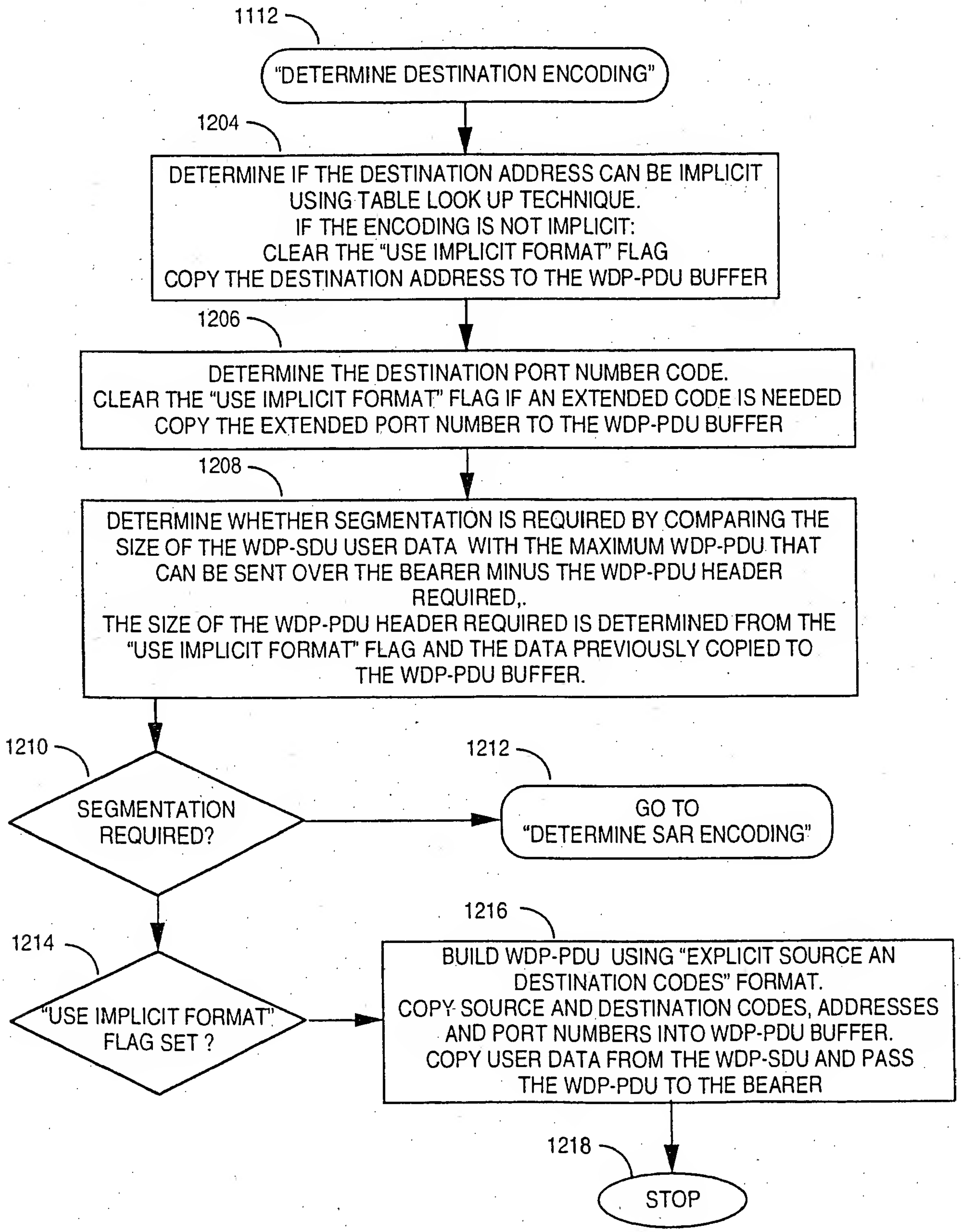
1000

FIG. 10


402
START
1104
SET "USE IMPLICIT FORMAT" FLAG
1106
DETERMINE RF BEARER (AND RECEIVER) CAPABILITY:
1) DESTINATION ADDRESS TYPE PASSED WITH WDP-SDU,
2) MAPPING DESTINATION ADDRESS TO RF BEARER ADDRESS
3) IMPLICITLY
1108
DETERMINE SOURCE ADDRESS TYPE:
1) SOURCE ADDRESS IS IMPLICIT
USE IMPLICIT SOURCE ADDRESS TYPE,
2) SOURCE ADDRESS NOT IMPLICIT
USE SOURCE ADDRESS TYPE FROM WDP-SDU,
COPY SOURCE ADDRESS INTO WDP-PDU BUFFER
CLEAR "USE IMPLICIT FORMAT" FLAG
1110
DETERMINE SOURCE PORT CODE:
1) SOURCE PORT NUMBER MATCHES ONE OF THE CONSTANT WAP
PORT NUMBERS DEFINED IN THE WDP PROTOCOL SPECIFICATION
USE THE ASSOCIATED SOURCE PORT CODE,
2) SOURCE CODE CAN BE IMPLICIT
USE THE IMPLICIT SOURCE PORT CODE,
3)SOURCE PORT NUMBER IS LESS THAN 25:
USE THE 1 BYTE EXTENDED CODE, OTHERWISE
USE THE 2 BYTE EXTENDED CODE,
4) EITHER THE 1 BYTE OR 2 BYTE EXTENDED CODE IS REQUIRED.
CLEAR THE "USE IMPLICIT FORMAT" FLAG.
COPY THE PORT NUMBER INTO THE WDP-PDU BUFFER
1112
GO TO
"DETERMINE DESTINATION ENCODING"

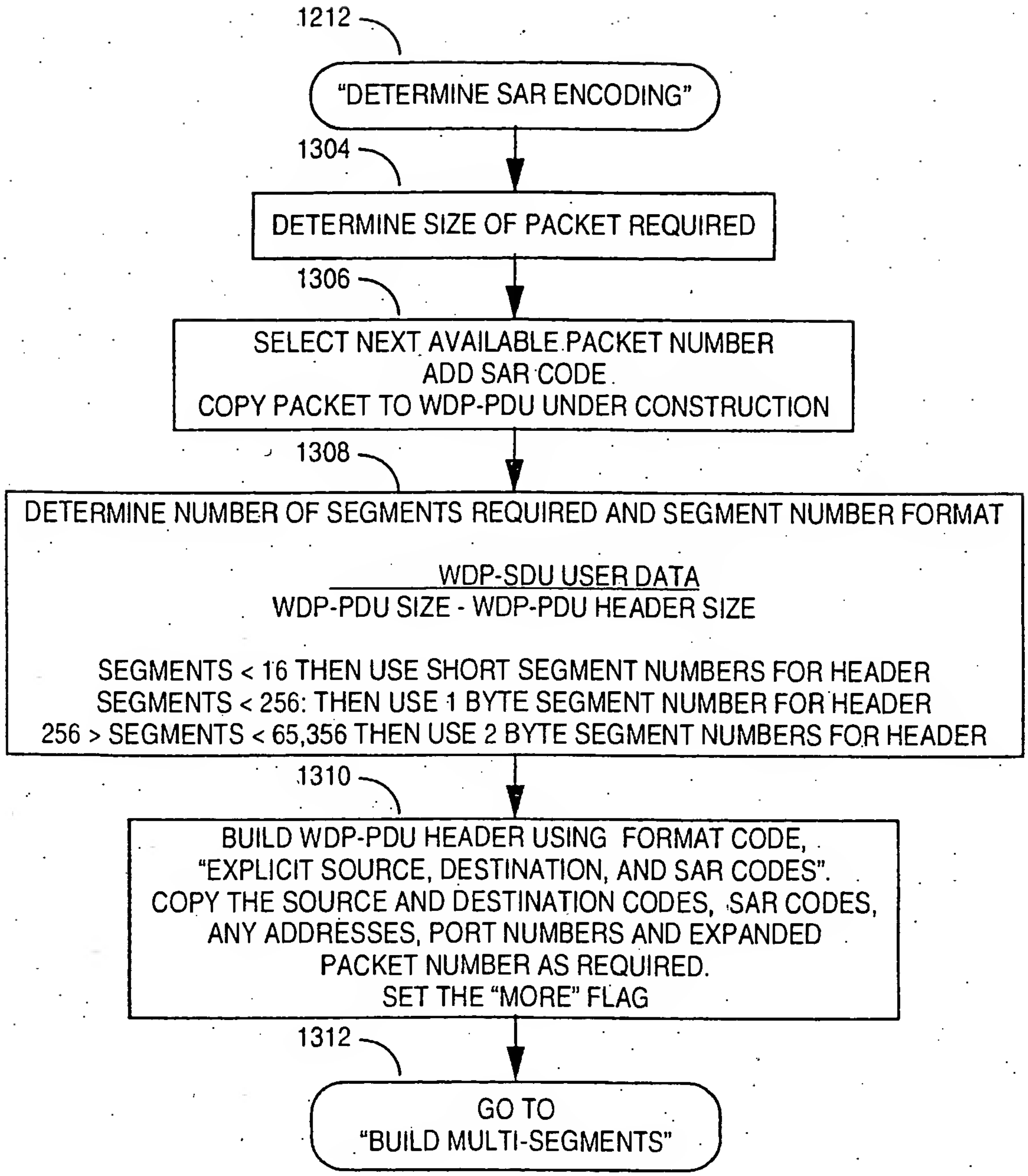
1100

FIG. 11


1112
"DETERMINE DESTINATION ENCODING"
1204
DETERMINE IF THE DESTINATION ADDRESS CAN BE IMPLICIT USING TABLE LOOK UP TECHNIQUE.
IF THE ENCODING IS NOT IMPLICIT:
CLEAR THE "USE IMPLICIT FORMAT" FLAG
COPY THE DESTINATION ADDRESS TO THE WDP-PDU BUFFER
1206
DETERMINE THE DESTINATION PORT NUMBER CODE.
CLEAR THE "USE IMPLICIT FORMAT" FLAG IF AN EXTENDED CODE IS NEEDED
COPY THE EXTENDED PORT NUMBER TO THE WDP-PDU BUFFER
1208
DETERMINE WHETHER SEGMENTATION IS REQUIRED BY COMPARING THE SIZE OF THE WDP-SDU USER DATA WITH THE MAXIMUM WDP-PDU THAT CAN BE SENT OVER THE BEARER MINUS THE WDP-PDU HEADER REQUIRED,.
THE SIZE OF THE WDP-PDU HEADER REQUIRED IS DETERMINED FROM THE "USE IMPLICIT FORMAT" FLAG AND THE DATA PREVIOUSLY COPIED TO THE WDP-PDU BUFFER.
1210
SEGMENTATION REQUIRED?
1212
GO TO "DETERMINE SAR ENCODING"
1216
BUILD WDP-PDU USING "EXPLICIT SOURCE AN DESTINATION CODES" FORMAT.
COPY SOURCE AND DESTINATION CODES, ADDRESSES AND PORT NUMBERS INTO WDP-PDU BUFFER.
COPY USER DATA FROM THE WDP-SDU AND PASS THE WDP-PDU TO THE BEARER
1214
"USE IMPLICIT FORMAT" FLAG SET ?
1218
STOP

1200

FIG. 12


1212
"DETERMINE SAR ENCODING"
1304
DETERMINE SIZE OF PACKET REQUIRED
1306
SELECT NEXT AVAILABLE PACKET NUMBER
ADD SAR CODE
COPY PACKET TO WDP-PDU UNDER CONSTRUCTION
1308
DETERMINE NUMBER OF SEGMENTS REQUIRED AND SEGMENT NUMBER FORMAT
WDP-SDU USER DATA
WDP-PDU SIZE - WDP-PDU HEADER SIZE
SEGMENTS < 16 THEN USE SHORT SEGMENT NUMBERS FOR HEADER
SEGMENTS < 256: THEN USE 1 BYTE SEGMENT NUMBER FOR HEADER
256 > SEGMENTS < 65,356 THEN USE 2 BYTE SEGMENT NUMBERS FOR HEADER
1310
BUILD WDP-PDU HEADER USING FORMAT CODE,
"EXPLICIT SOURCE, DESTINATION, AND SAR CODES".
COPY THE SOURCE AND DESTINATION CODES, SAR CODES,
ANY ADDRESSES, PORT NUMBERS AND EXPANDED
PACKET NUMBER AS REQUIRED.
SET THE "MORE" FLAG
1312
GO TO
"BUILD MULTI-SEGMENTS"

1300

FIG. 13

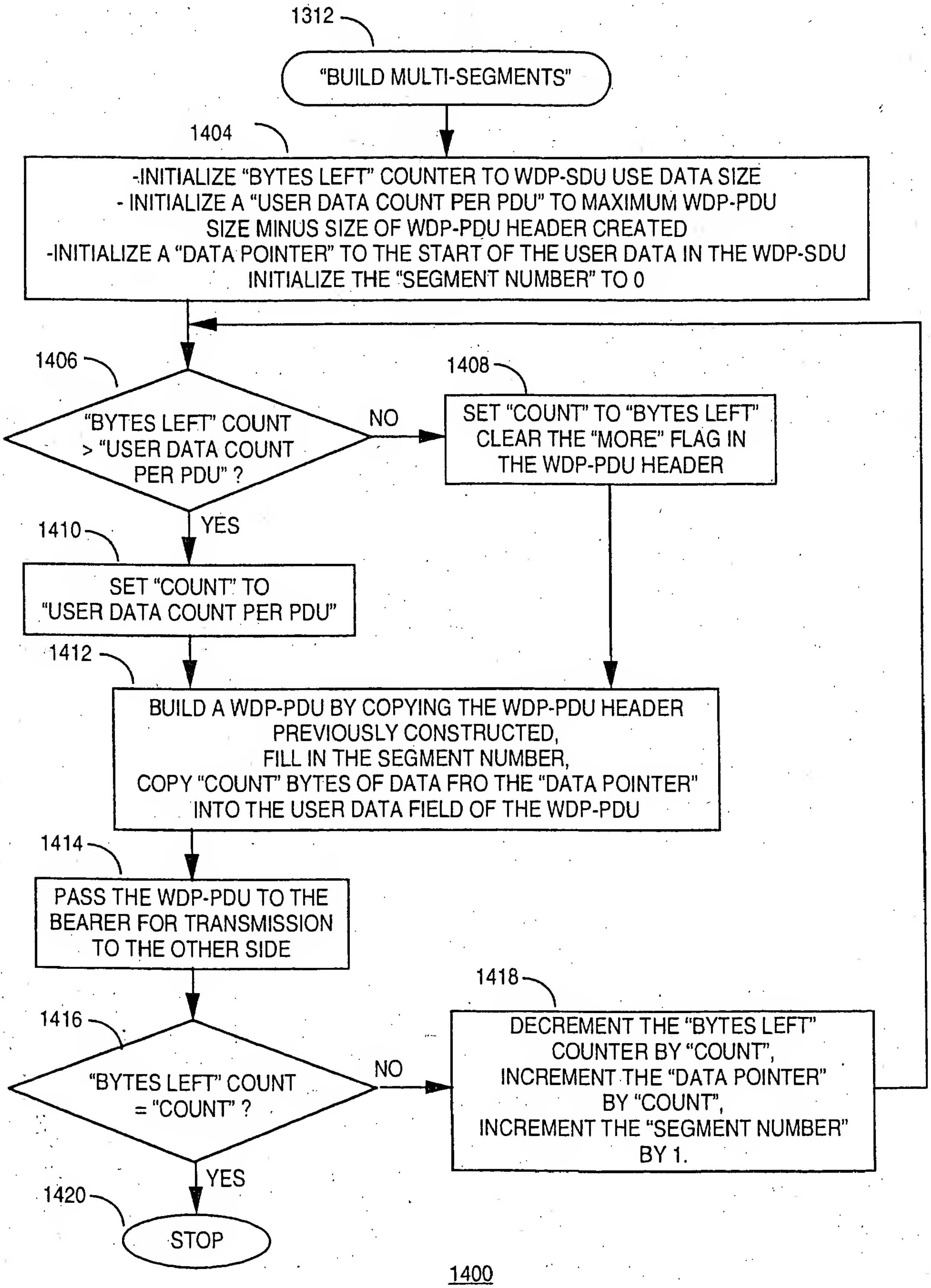

1312
"BUILD MULTI-SEGMENTS"
1404
-INITIALIZE "BYTES LEFT" COUNTER TO WDP-SDU USE DATA SIZE
- INITIALIZE A "USER DATA COUNT PER PDU" TO MAXIMUM WDP-PDU SIZE MINUS SIZE OF WDP-PDU HEADER CREATED
-INITIALIZE A "DATA POINTER" TO THE START OF THE USER DATA IN THE WDP-SDU
INITIALIZE THE "SEGMENT NUMBER" TO 0
1406
"BYTES LEFT" COUNT > "USER DATA COUNT PER PDU" ?
NO
1408
SET "COUNT" TO "BYTES LEFT" CLEAR THE "MORE" FLAG IN THE WDP-PDU HEADER
YES
1410
SET "COUNT" TO "USER DATA COUNT PER PDU"
1412
BUILD A WDP-PDU BY COPYING THE WDP-PDU HEADER PREVIOUSLY CONSTRUCTED,
FILL IN THE SEGMENT NUMBER,
COPY "COUNT" BYTES OF DATA FRO THE "DATA POINTER" INTO THE USER DATA FIELD OF THE WDP-PDU
1414
PASS THE WDP-PDU TO THE BEARER FOR TRANSMISSION TO THE OTHER SIDE
1418
DECREMENT THE "BYTES LEFT" COUNTER BY "COUNT",
INCREMENT THE "DATA POINTER" BY "COUNT",
INCREMENT THE "SEGMENT NUMBER" BY 1.
1416
"BYTES LEFT" COUNT = "COUNT" ?
NO
YES
1420
STOP
1400

FIG. 14

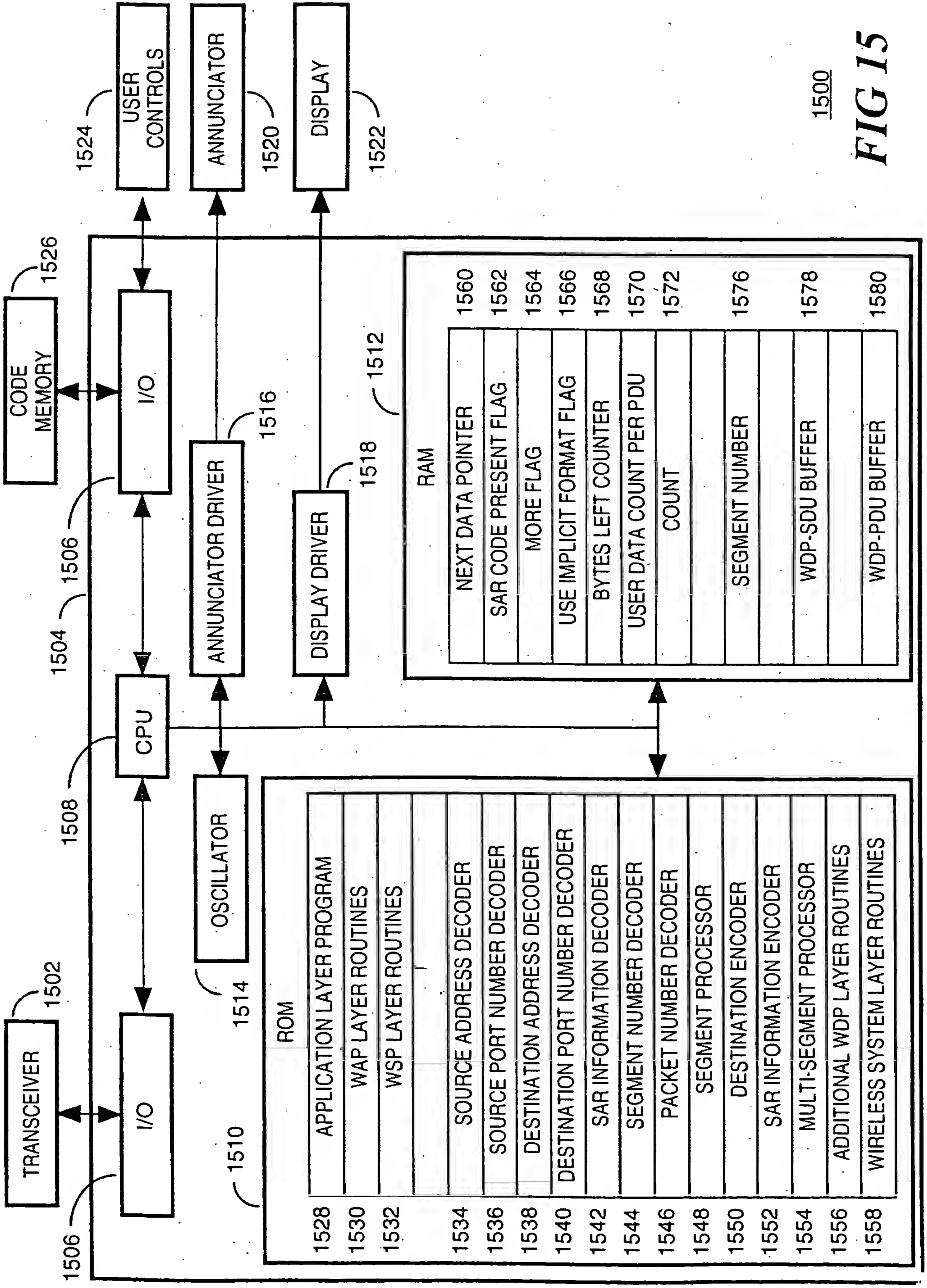

1500
TRANSCEIVER
1502
I/O
1506
CODE MEMORY
1526
I/O
1506
1504
CPU
1508
USER CONTROLS
1524
ANNUNCIATOR
1520
DISPLAY
1522
ANNUNCIATOR DRIVER
1516
DISPLAY DRIVER
1518
OSCILLATOR
1514
ROM
1510
1528 APPLICATION LAYER PROGRAM
1530 WAP LAYER ROUTINES
1532 WSP LAYER ROUTINES
1534 SOURCE ADDRESS DECODER
1536 SOURCE PORT NUMBER DECODER
1538 DESTINATION ADDRESS DECODER
1540 DESTINATION PORT NUMBER DECODER
1542 SAR INFORMATION DECODER
1544 SEGMENT NUMBER DECODER
1546 PACKET NUMBER DECODER
1548 SEGMENT PROCESSOR
1550 DESTINATION ENCODER
1552 SAR INFORMATION ENCODER
1554 MULTI-SEGMENT PROCESSOR
1556 ADDITIONAL WDP LAYER ROUTINES
1558 WIRELESS SYSTEM LAYER ROUTINES
RAM
1512
NEXT DATA POINTER 1560
SAR CODE PRESENT FLAG 1562
MORE FLAG 1564
USE IMPLICIT FORMAT FLAG 1566
BYTES LEFT COUNTER 1568
USER DATA COUNT PER PDU 1570
COUNT 1572
SEGMENT NUMBER 1576
WDP-SDU BUFFER 1578
WDP-PDU BUFFER 1580

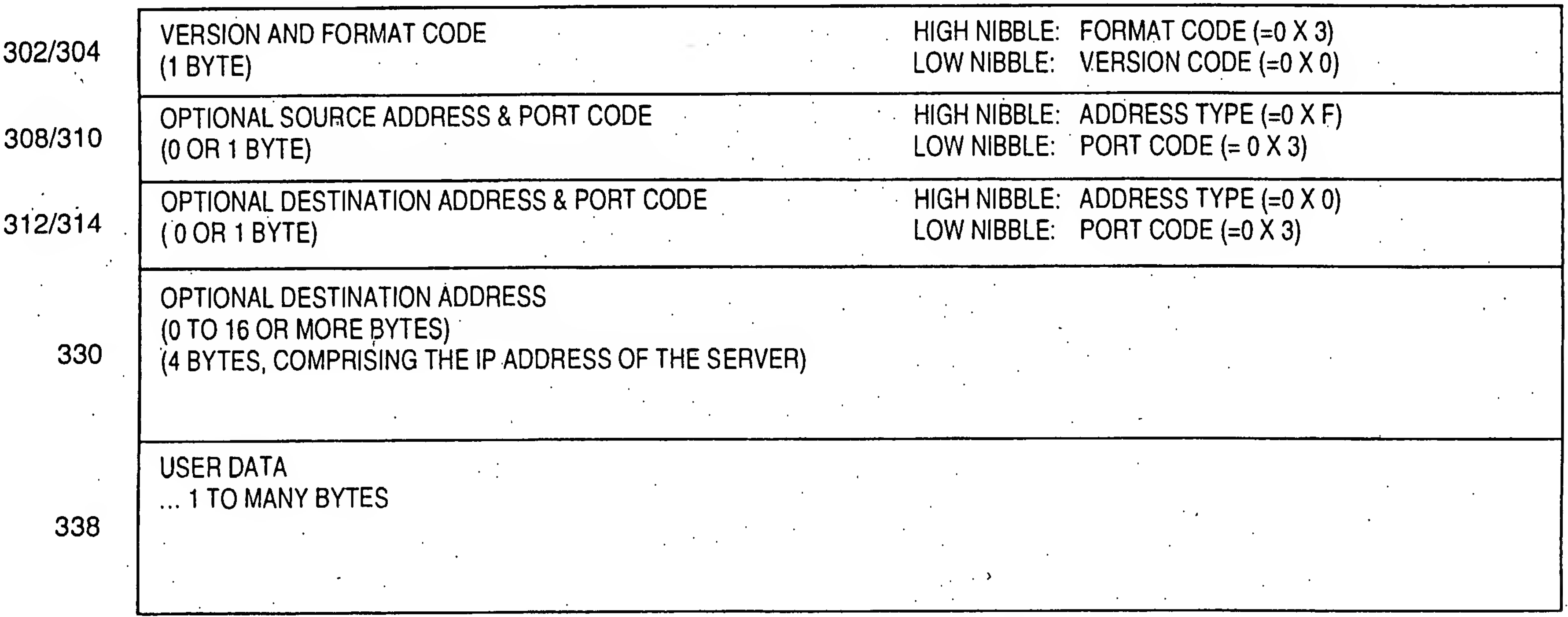
FIG 15

302/304	VERSION AND FORMAT CODE (1 BYTE)	HIGH NIBBLE: FORMAT CODE (=0 X 2) LOW NIBBLE: VERSION CODE (=0 X 0)
338	USER DATA ... 1 TO MANY BYTES	

FIG. 16

302/304	VERSION AND FORMAT CODE (1 BYTE)	HIGH NIBBLE: FORMAT CODE (=0 X 3) LOW NIBBLE: VERSION CODE (=0 X 0)
308/310	OPTIONAL SOURCE ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: ADDRESS TYPE (=0 X 0) LOW NIBBLE: PORT CODE (= 0 X 3)
312/314	OPTIONAL DESTINATION ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: ADDRESS TYPE (=0 X F) LOW NIBBLE: PORT CODE (=0 X 3)
324	OPTIONAL SOURCE ADDRESS (0 TO 16 OR MORE BYTES) (4 BYTES, COMPRISING THE IP ADDRESS OF THE SERVER)	
338	USER DATA ... 1 TO MANY BYTES	

FIG. 17



302/304	VERSION AND FORMAT CODE (1 BYTE)	HIGH NIBBLE: FORMAT CODE (=0 X 3) LOW NIBBLE: VERSION CODE (=0 X 0)
308/310	OPTIONAL SOURCE ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: ADDRESS TYPE (=0 X F) LOW NIBBLE: PORT CODE (= 0 X 3)
312/314	OPTIONAL DESTINATION ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: ADDRESS TYPE (=0 X 0) LOW NIBBLE: PORT CODE (=0 X 3)
330	OPTIONAL DESTINATION ADDRESS (0 TO 16 OR MORE BYTES) (4 BYTES, COMPRISING THE IP ADDRESS OF THE SERVER)	
338	USER DATA ... 1 TO MANY BYTES	

FIG. 18

302/304	VERSION AND FORMAT CODE (1 BYTE)	HIGH NIBBLE: LOW NIBBLE:	FORMAT CODE (=0 X 3) VERSION CODE (=0 X 0)	
308/310	OPTIONAL SOURCE ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: LOW NIBBLE:	ADDRESS TYPE (=0 X 0) PORT CODE (= 0 X 3)	
312/314	OPTIONAL DESTINATION ADDRESS & PORT CODE (0 OR 1 BYTE)	HIGH NIBBLE: LOW NIBBLE:	ADDRESS TYPE (=0 X F) PORT CODE (=0 X 3)	
316	OPTIONAL SEGMENT & RE-ASSEMBLY CODE (0 OR 1 BYTE)	HIGH NIBBLE: LOW NIBBLE:	MORE FLAG + SEGMENT NUMBER CODE PACKET NUMBER CODE	318/320
324	OPTIONAL SOURCE ADDRESS (0 TO 16 OR MORE BYTES) (4 BYTE IPV4 ADDRESS OF THE SERVER)			
326	OPTIONAL SOURCE PORT NUMBER (2 BYTES)			
330	OPTIONAL DESTINATION ADDRESS (4 BYTE DATA TAC LLI))			
332	OPTIONAL DESTINATION PORT NUMBER (2 BYTES)			
336	OPTIONAL EXTENDED PACKET NUMBER (0, 1, 2, OR 4 BYTES)			
338	USER DATA ... 1 TO MANY BYTES			

FIG. 19

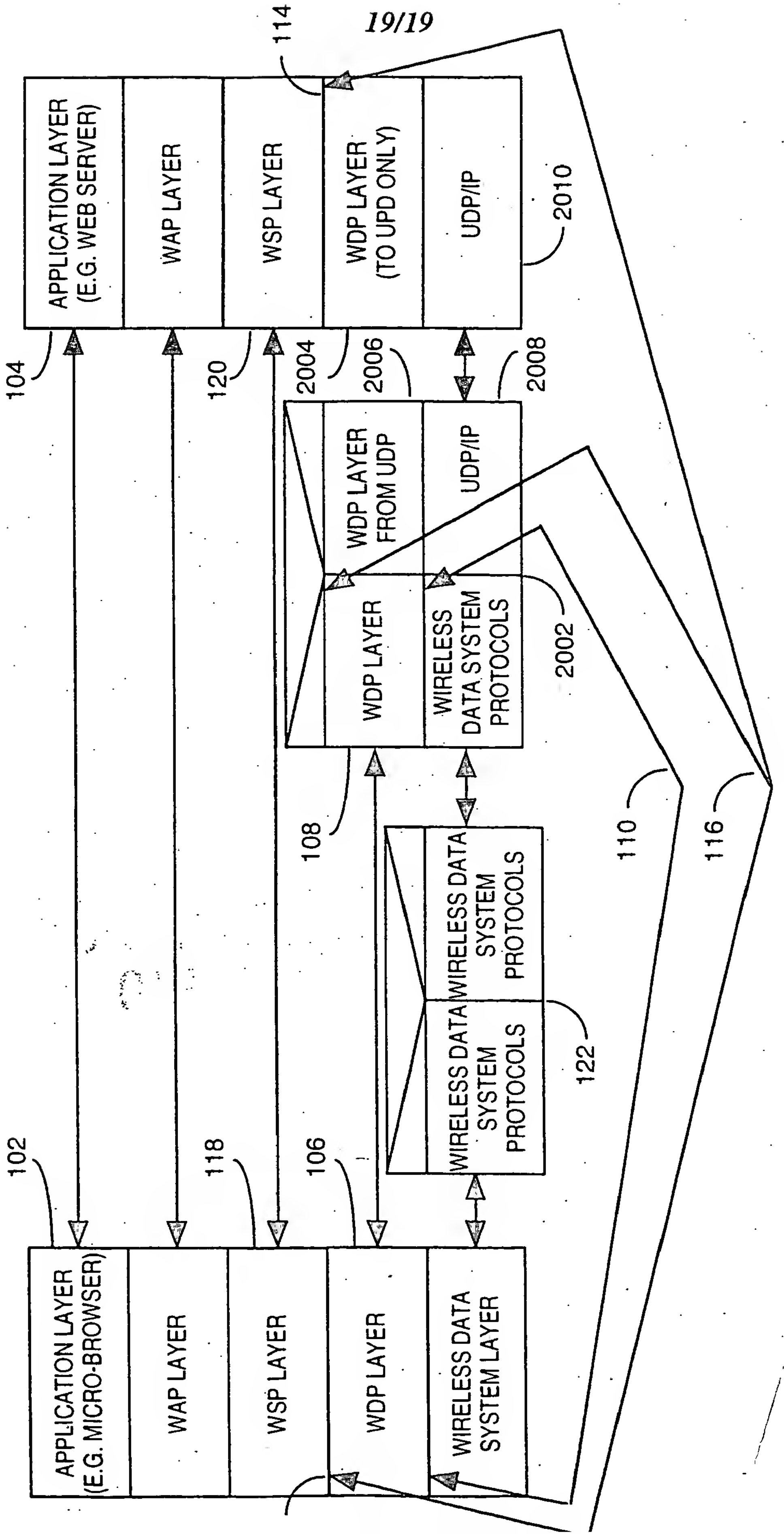

APPLICATION LAYER (E.G. WEB SERVER)
WAP LAYER
WSP LAYER
WDP LAYER (TO UPD ONLY)
UDP/IP
114
2010
104
120
2004
2006
2008
WDP LAYER FROM UDP
UDP/IP
WDP LAYER
WIRELESS DATA SYSTEM PROTOCOLS
2002
108
110
116
WIRELESS DATA SYSTEM PROTOCOLS
WIRELESS DATA SYSTEM PROTOCOLS
122
102
118
106
APPLICATION LAYER (E.G. MICRO-BROWSER)
WAP LAYER
WSP LAYER
WDP LAYER
WIRELESS DATA SYSTEM LAYER
2000

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US99/20748

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/30
US CL :370/230,349,401,471,474
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 370/230,349,401,471,474, 913; 709/203, 236;

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

West; wireless; datagram; pdu; and header

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,724,515 A (BARNES et al.) 03 March 1998, Fig. 5 and ig. 6.	1-4
A	US 5,742,611 A (BRANDIN) 21 April 1998, Fig. 2 and Fig. 3.	1-4
A	US 5,793,758 A (PENNERS) 11 August 1998, Fig.3.	1-4
Y	US 5,627,829 A (GLEESON et al.) 06 May 1997Fig.12A.	1-4
Y	US 5,446,736 A (GLEESON et al.) 29 August 1995, Fig 12A.	1-4

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search 03 DECEMBER 1999	Date of mailing of the international search report 14 JAN 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer SEEMA S. RAO Telephone No. (703) 308-3900